+50 DICAS PARA

Editora Aleph

ROTINAS E TRUQUES COMENTADOS

+5Ø DICAS PARA MSX

ROTINAS E TRUQUES COMENTADOS

Aldo Barduco Jr.
Henrique de Figueredo Luz
Milton Maldonado Jr.
Pierluigi Piazzi
Renato da Silva Oliveira

2ª EDIÇÃO
1989

AGRADECIMENTOS

Todos os textos deste livro foram digitados em micros Expert e HOTBIT gentilmente cedidos pela Gradiente e Sharp, usando o editor de textos MSX-WRITE fornecido pela XSW em conjunto com um filtro de fontes alternativas desenvolvido por esta soft-house.

A composição do texto e as ilustrações técnicas foram feitos em impressoras MÔNICA PLUS e OLIVIA gentilmente cedidas pela ELEBRA.

Queremos expressar nossos agradecimentos a essas empresas por colaborarem na elaboração de mais uma obra dedicada aos usuários de MSX no Brasil.

EXPEDIENTE:
Coordenação Editorial: Pierluigi Piazzi
Coordenação Didática: Betty Fromer Piazzi
Produção Editorial: Rosa Kogan Fromer
Editoração: Henrique de Figueredo Luz
Arte e Capa: Ana Lúcia Antico & Nicoletti

ALEPH
Publicações e Assessoria Pedagógica Ltda.
Rua Heitor de Souza Pinheiro 35 -3º A
05750 São Paulo - SP (011) 843-3202
Caixa Postal 20.707 - CEP 01498

Dados de Catalogação na Publicação (CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

Piazzi, Pierluigi, 1943-
P647m + 5Ø dicas para MSX : rotinas e truques comentados / Piazzi e Cia. -- São Paulo : Aleph, 1989. -- (Coleção MSX)

Bibliografia.

1. MSX (Computadores) - Programação 2. MSX-BASIC (Linguagem de programação para computadores) I. Título. II. Série.

89-0006 CDD-001.6424
-001.642

Índices para catálogo sistemático:
1. MSX : Computadores : Programação : Processamento de dados 001.642
2. MSX : Linguagem de programação : Computadores : Processamento de dados 001.6424

SUMARIO

DICAS PARA O DRIVE

DICAS DE PERIFÉRICOS

APÊNDICES

NOTA DO EDITOR

A Editora Aleph está publicando livros para MSX desde o lançamento deste padrão no Brasil. A rigor começamos até um pouco antes, pois os manuais do Expert foram produzidos por nossa equipe: quando o 1º MSX chegou na primeira loja brasileira, dentro da caixa já estavam livros por nós editados!

Desde então procuramos alimentar um público sedento de informações. Neste processo tivemos que criar, praticamente a partir do nada, usando autores brasileiros escrevendo para um público brasileiro

Foi um processo mais penoso do que o ocorrido com o Apple ou o Sinclair: obter informações a partir de uma fonte americana ou inglesa é tremendamente mais fácil do que romper a barreira linguística e cultural de uma fonte japonesa!

Foi mais penoso mas muito mais gratificante: o processo de criação é muito mais enriquecedor do ponto de vista intelectual (financeiro que não!) do que cópia pura e simples.

Aliás é isso que os ferrenhos defensores de reserva de mercado não conseguem entender.

Para quem não sabe, Reserva de Mercado na área de Micro-Informática é um mecanismo surrealista concebido para encher os bolsos de piratas, industrias incompetentes, contrabandistas e alguns fiscais alfandegários corruptos.

É de domínio público o fato de que o maior fornecedor de micro-informática no Brasil é o "importabando"!

Trata-se do típico caso de "solução" pior que o problema: isso já aconteceu nos Estados Unidos na época da Lei-Seca: ao invés de acabar com o alcoolátras, criaram os "gangsters"!

No mundo do MSX, porém toda essa sujeira (ou quase toda) passa longe.

Temos um mercado de brasileiros, suprido por fabricantes brasileiros, onde ainda se pode trabalhar num clima em que "idealismo" não signifique "bancarrota".

E é nesse clima que nós da Aleph estamos trabalhando. Ao longo desses anos estabelecemos um diálogo com os nossos leitores (alguns dos quais se divertem mais lendo essas "notas do editor" do que

digitando os programas do livro!) que permitiu detectarmos o fato de que estamos no caminho certo. E quando não estamos ,também temos a resposta na hora, de maneira a podermos efetuar prontas "correções de rumo".

Quando publicamos o livro "100 dicas para MSX" recebemos dois retornos extremamente significativos:um elogioso e outro crítico.

O elogioso resume-se no sucesso de vendas, nas cartas e telefonemas de apoio e no pedido de mais dicas (donde este "+50...").

O crítico foi feito por uma parcela de eleitores apresentando queixas que, sintetizados, diziam "gostei das dicas, sei usá-las mas não me foi explicado o porquê de seu funcionamento!".

Essa calou-fundo! Afinal de contas nosso carro-chefe sempre foi o aspecto didático das nossas obras!.

Como "correção de rumo", então, resolvemos colocar menos dicas neste livro mas explicá-las mais exaustivamente.

Quando o programa apresentado está em BASIC, sua explicação está no próprio texto da dica. Quando ele está em Linguagem de Máquina, o programa-fonte em Assembly foi listado e comentado no apêndice 1.

Desta forma tentamos atender aquele leitor mais exigente que quer saber tudo nos seus "mííínimos detalhes"!

Mais um assunto: muitos leitores nos telefonam ou escrevem com dúvidas (essa, aliás, é nossa maior fonte de informações sobre as necessidades do nosso público). Ao tentar orientar o "consulente", muitos vezes o remetemos a uma certa página de um dado livro e recebemos a cândida resposta: "essa página não tenho porque xeroquei apenas um pedaço do livro"!

Não vamos discutir esse papo de xerox do ponto de vista ético (afinal cópia não autorizada também é pirataria!) mas de um ponto de vista bem pragmático: o interesse do público leitor.

Se nossos livros (e de outras editoras) não forem comprados mas sim xerocados, a tiragem será pequena. Com tiragens pequenas o custo unitário do livro aumenta (existe um custo fixo que é o mesmo para qualquer tiragem), e as vendas diminuem.

O leitor é prejudicado porque paga os livros mais caros, o editor é prejudicado e tem menos estímulo para lançar novas obras e, mais grave de tudo, o autor, um brasileiro que deu um duro danado, passando

noites em claro para escrever informações úteis a todo um universo de leitores, passa a receber uma quantia irrisória de Direitos Autorais (o cheque que eu, como editor, assino com mais prazer).

Resultado: mais desestímulo e mais usuários que sub-utilizam seu equipamento por disporem de informações mais escassas e mais caras!

Por isso vou parafrasear aquele locutor mambembe da Rádio Camanducaia. "E agora um apelo: pelo amor de Deus, anuncie em nossa emissora"!

Pois é leitor: "Pelo amor de Deus,COMPRE livros"! Não faça cópias de pedaços sob pena de quebrar toda uma corrente de trabalho que bem ou mal funciona, uma das raras coisas que funcionam nesse País.

Enquanto estou escrevendo esta Nota do Editor, tenho à minha frente uma pasta repleta de cartas de editores estrangeiros que me oferecem os direitos de tradução por algumas centenas de dólares (muito menos do que pago a meus autores brasileiros).

Às vezes me pergunto se não seria mais fácil, lucrativo e simples apelar para este caminho.

Depois penso no Brasil que meus filhos vão herdar: não gostaria que isso aqui voltasse a ser colônia.

Guardo a pasta das traduções, pego na caneta e continuo minhas notas!

BIBLIOGRAFIA REMISSIVA

Para melhor compreensão das dicas deste livro e um maior aprofundamento dos assuntos abordados, relacionamos a seguir uma série de títulos já publicados pela ALEPH para microcomputadores da linha MSX.

Após cada título, listamos o número das dicas relacionadas com os assuntos abordados na obra.

APROFUNDANDO-SE NO MSX - 1, 3, 4, 5, 6, 8, 10, 11, 12, 13, 17, 36, 37, 46, 49
PROGRAMAÇÃO AVANÇADA EM MSX - 14, 15, 20, 29, 30, 31, 32, 33
LINGUAGEM BASIC MSX - 2, 13, 21, 23, 24, 25, 26, 50
DRIVE LEOPARD DE 3 1/2' - 13, 40, 41, 42, 43, 44, 45, 46, 48
HOTLOGO - 39
HOTWORD - 43
100 DICAS PARA MSX - 9, 12, 16, 29, 30, 47
CURSO DE MÚSICA PARA MSX - 19
COLEÇÃO DE PROGRAMAS Vol 2 - 11
CURSO DE BASIC - 38

Para os que pretendem avançar no domínio da linguagem BASIC, aconselhamos a leitura do PROGRAMAÇÃO PROFISSIONAL EM BASIC.

Se você quiser receber gratuitamente o Boletim Informativo da ALEPH contendo informações úteis sobre o padrão MSX, envie seu nome e endereço para: ALEPH - C.P. 20.707 CEP- 01498 S.Paulo - SP

PAGINAÇÃO DE TEXTO E GRÁFICOS

1

Ao fazermos um programa que se utilize da SCREEN 2 para fazer gráficos, surgem alguns inconvenientes a que os usuários de MSX estão até acostumados.

A impressão de caracteres na SCREEN 2 é lenta. Para imprimir novas informações em cima de algo já escrito é necessário, primeiramente, apagar a antiga informação. E finalmente o mais chato de tudo é que para cada linha de texto que precisamos usar, perdemos mais e mais pontos que poderiam ser usados na própria confecção do gráfico.

Quando estamos trabalhando em SCREEN 2, quase toda a VRAM é ocupada por por tabelas que definem a tela. Veja na figura 1.1 a distribuiçao das tabelas pela VRAM

Figura x.1

BASE	TABELA DE:	INÍCIO-FIM
12	Formação de caracteres	0000- 6143
10	Código dos caracteres	6144- 6911
13	atributos de sprites	6912- 7040
11	cores	8192-14335
14	formação dos sprites	14336-16383

Se você notou bem, a VRAM está praticamente lotada quando trabalhamos na SCREEN 2. Porém, entre as tabelas de atributos de sprites e a tabela de cores existe uma "janela" que não é usada. Note que uma tabela termina em 7040 e a outra só tem início em 8192.

Temos ,então, 1152 bytes da VRAM para dispormos como quisermos.

A organização de SCREEN 0 é muito mais simples e ocupa muito menos memória do que a SCREEN 2. Veja na figura 1.2 essa organização:

Figura 1.2

BASE	TABELA DE:	INÍCIO-FIM
0	Códigos impressos na tela	0000- 959
2	Formação dos caracteres	2048- 4095

São duas tabelas, uma de 960 bytes (0 a 959) e outra de 2048 bytes (2048 a 4095).

Isto é, tranqüilamente, muito mais memória do que dispomos na VRAM quando trabalhamos em SCREEN 2. Porém, temos na VRAM uma tabela que é relativamente pouco usada - a tabela de formação dos sprites.

Se transferirmos a tabela de formação dos caracteres para a tabela de formação dos sprites teremos 256 padrões pré-definidos para usarmos como sprites (Veja a dica número 5). Mas o mais importante, agora, é que conseguimos espaço para termos as tabelas da SCREEN 0 e SCREEN 2 ao mesmo tempo na VRAM.

Para alterarmos as tabelas da SCREEN 0 para o endereço que queremos, usamos o comando BASE da seguinte forma:

```
BASE (0)=7168
```

para que a tabela dos códigos dos caracters caia em uma área livre da VRAM (quando em SCREEN 2); e para a tabela de formação dos caracteres cair sobre a tabela de formação dos sprites usamos:

```
BASE (2)=BASE (14)
```

Precisamos, agora, de alguma meneira para mudar de screen sem usar o comando SCREEN do BASIC, pois quando executado ele inicializa todas as tabelas da VRAM.

Existem duas rotinas do BIOS que apenas setam o modo de operação do VDP. São elas:

SETTXT &H0078
SETGRP &H007E

A SETTXT ajusta o VDP para modo TeXTo e a SETGRP para modo gráfico (GRaPhic).

A rotina SETGRP, sempre que chamada, muda o registro 7 do VDP, no qual estão definidas a cor dos caracteres e a cor de fundo no modo SCREEN 0. Devido à essa característica é necessário guardar o registro 7 do VDP em uma variável antes de chamar a rotina SETGRP da seguinte forma:

variável=VDP(7)

e ao chamar a rotina SETTXT comandar:

VDP(7)=variável

Para que os comandos funcionem corretamente em cada screen é necessário dizer ao interpretador em qual delas estamos trabalhando. Fazemos isso através da variável de sistema SCRMOD (&HFCAF), POKEando 0 ou 2, de acordo com a screen que queremos usar.

Digite o programa da figura 1.3 e veja seu efeito.

Figura 1.3

```
100 KEY1,"      X="
110 KEY3,"      Y="
120 KEY5,"                 "
130 DEFUSR=&H78 :'screen 0
140 DEFUSR1=&H7E:'screen 2
145 VP=VDP(7)
150 BASE(0)=7168
160 BASE(2)=BASE(14)
170 SCREEN 0
180 SCREEN 2
190 FOR R=0 TO 255 STEP 3
200 POKE 0,USR1(0)
210 POKE &HFCAF,2
220 Y1=INT(96+90*SIN(R/64*3.14))
230 LINE (R-2,Y1)-(R,Y1+2),,B
240 FOR L=0TO 500:NEXT L
250 POKE 0,USR0(0)
260 POKE &HFCAF,0
270 VDP(7)=VP
280 PRINT"(";:PRINTUSING"###";R;:PRINT","
;:PRINTUSING"###";Y1;:PRINT")  ";
290 KEY2,STR$(R)
300 KEY4,STR$(Y1)
```

```
310 FOR L=0TO 500:NEXT L
320 NEXT
330 A$=INPUT$(1)
340 IF T=0 THEN T=1:POKE 0,USR1(0) ELSE T
=0:POKE 0,USR(0)
350 POKE &HFCAF,T
360 VDP(7)=VP
370 GOTO 330
```

Note que inicialmente são dados os comandos DEFUSR para DEFinir o ponto de entrada das rotinas SETTXT e SETGRP (linhas 130 e 140).

Em seguida o registro 7 do VDP é armazenado (linha 145).

Nas linhas 150 e 160 mudamos o endereço das tabelas da SCREEN 0 na VRAM.

O comando SCREEN 0 da linha 170 serve para que essas tabelas sejam inicializadas e transferidas para a VRAM.

O comando SCREEN 2 (linha 180) inicializa as tabelas da SCREEN 2.

Temos, então, a seguinte seqüência de comandos para inicializarmos a VRAM para as duas screens:

```
DEFUSR=&H78 :'screen 0
DEFUSR1=&H7E:'screen 2
VP=VDP(7)
BASE(0)=7168
BASE(2)=BASE(14)
SCREEN 0
SCREEN 2
```

Antes de serem dados os comandos específicos da tela gráfica devemos ter as instruções:

```
POKE 0,USR1(0)
POKE &HFCAF,2
```

e para comandarmos a tela de texto a sequencia:

```
POKE 0,USR0(0)
POKE &HFCAF,0
VDP(7)=VP
```

EDITOR DE DRAWS

2

O comando DRAW utiliza uma série de sub-comandos (ou macro-comandos) que praticamente formam uma sub-linguagem gráfica dentro da linguagem BASIC.

Esses subcomandos devem estar em uma "STRING", ou seja, entre aspas ou em uma variável alfanumérica.

Este EDITOR visa facilitar o uso dessa instrução gerando um arquivo de linhas "DRAW" que poderá ser gravado e depois MERGEado (fazendo um MERGE) para o uso em outros programas.

Comandos.

[F1] - Define o início do traço e marca com uma circunferência vermelha.
[F2] - Desenha um risco até o cursor.
[F3] - Apaga o ultimo traço feito.
[F4] - Transforma o que estava na tela em linha "DRAW" e permite a gravação ou exibição na tela.
(HOME) - Apaga a tela e o desenho (da MEMóRIA).
As teclas de setas movem o cursor.

Funcionamento do programa.

Da linha 50 a 220: inicialização.
Da 230 a 350: processamento.
Da 360 a 1210: sub-rotinas das teclas de função.
Da 1220 em diante: sub-rotinas auxiliares.

Rotina de TRATAMENTO DE T$: Procura o último sub-comando da instrução DRAW quando a string ultrapassa os 230 caracteres.

Rotina de "BACK-INSTR": Faz como a instrução INSTR do Basic só que de trás para frente.

Rotina de "BACK-DRAW": Inverte os valores para que se desenhe para trás.

Uma idéia é fazer sua assinatura para colocá-la em uma apresentação, personalizando seus programas (ver pág. 76 do livro Coleção de Programas vol. 1).

```
10 '        +--------------------+
20 '        ! # OS ITALIANOS # !
30 '        +--------------------+
40 '
```

```
50 DEFINT A-Z:CLEAR 1000,&H9C3F
60 DI=&H9C3F:KEY OFF:WIDTH40
70 CLS:COLOR 15,1,1:LOCATE 11,0:PRINT"ED
ITOR DE DRAWS":PRINT STRING$(40,"-")
80 LOCATE 0,3:INPUT"NOME DO ARQUIVO ";WW
$:IF LEFT$(WW$,1)<"A" THEN 80
90 N$=WW$
100 LOCATE 0,5:INPUT "LINHA INICIAL";LI:
IF LI<0 OR LI>60000! THEN 100
110 SCREEN 2
120 GOSUB 390
130 GOSUB 470
140 GOSUB 520
150 ON KEY GOSUB 570,700,810,880
160 '--------------------------------
170 ' ATIVA AS TECLAS DE FUNCAO
180 '--------------------------------
190 FOR I=1 TO 4
200 KEY(I) ON
210 NEXT I
220 X1=0:Y1=0:XU=0:YU=0:KEY (2) OFF
230 DRAW"S32"
240 PUT SPRITE 1,(8*X1+13,8*Y1+12),12,1
250 PUT SPRITE 2,(8*X2+13,8*Y2+12),L,2
260 A=STICK(0)ORSTICK(1)ORSTICK(2)
270 K$=INKEY$
280 IF K$=CHR$(11) THEN DI=&H9C3F:GOSUB
370
290 X1=X1-(A=2)-(A=3)-(A=4)+(A=6)+(A=7)+
(A=8)
300 Y1=Y1-(A=4)-(A=5)-(A=6)+(A=8)+(A=1)+
(A=2)
310 IF X1<0 THEN X1=29
320 IF X1>29 THEN X1=0
330 IF Y1<0 THEN Y1=21
340 IF Y1>21 THEN Y1=0
350 IF L<>8 THEN KEY(2) OFF ELSE KEY(2)
ON
360 GOTO 240
370 CLS
380 '-------------------------------
390 ' RETICULA A SCREEN 2
400 '-------------------------------
410 FOR C%=16 TO 255 STEP 8
420 FOR L%=16 TO 191 STEP 8
430 PSET(C%,L%)
440 NEXT L%,C%
450 RETURN
```

```
460 '--------------------------------
470 'DEFINE SPRITE VERDE
480 '--------------------------------
490 SPRITE$(1)=CHR$(16)+CHR$(16)+CHR$(16
)+CHR$(238)+CHR$(16)+CHR$(16)+CHR$(16)+C
HR$(0)
500 RETURN
510 '--------------------------------
520 'DEFINE SPRITE VERMELHO
530 '--------------------------------
540 SPRITE$(2)=CHR$(56)+CHR$(124)+CHR$(2
54)+CHR$(238)+CHR$(254)+CHR$(124)+CHR$(5
6)+CHR$(0)
550 RETURN
560 '--------------------------------
570 ' ROTINA DE F1
580 '--------------------------------
590 L=8:B$="B":GOSUB 1240
600 A$=""
610 PSET(XU*8+16,YU*8+16)
620 XA=X1-XU:YA=Y1-YU
630 X$=STR$(XA):IF MID$(X$,1,1)=" " THEN
  X$="+"+MID$(X$,2,LEN(X$))
640 Y$=STR$(YA):IF MID$(Y$,1,1)=" " THEN
  Y$="+"+MID$(Y$,2,LEN(Y$))
650 A$=B$+"M"+X$+","+Y$
660 DRAW"XA$;"
670 FOR F=1 TO LEN(A$):L$=MID$(A$,F,1):P
OKE DI+F,ASC(L$):NEXT:DI=DI+LEN(A$)
680 KEY(2) ON:RETURN
690 '--------------------------------
700 ' ROTINA DE F2
710 '--------------------------------
720 IF X1=X2 AND Y1=Y2 THEN RETURN
730 KEY(2) OFF:B$=""
740 XU=X1:YU=Y1
750 PSET(X2*8+16,Y2*8+16)
760 GOSUB 1130
770 L=0
780 KEY(1) ON
790 RETURN
800 '--------------------------------
810 ' ROTINA DE F3
820 '--------------------------------
830 GOSUB 1360
840 RETURN
850 '--------------------------------
860 ' ROTINA DE F4
```

```
870 '-----------------------------------
880 IF DI=&H9C3F THEN RETURN
890 SCREEN0:LOCATE 13,0:PRINT"EDITOR DE
DRAWS":PRINT STRING$(40,"-"):FOR A=1 TO
4:KEY(A)OFF:NEXT A
900 PRINT:PRINT"(1)-GRAVA EM DISCO":PRIN
T"(2)-GRAVA EM FITA":PRINT"(3)-MOSTRA NA
 TELA":PRINT"(4)-VOLTA A EDICAO":PRINT
910 A$=INKEY$
920 IF A$="1" THEN D$="A:":GOTO 970
930 IF A$="2" THEN D$="CAS:":GOTO 970
940 IF A$="3" THEN D$="CRT:":GOTO 970
950 IF A$="4" THEN RETURN 110
960 GOTO 910
970 PRINT:PRINT"##  EXECUTANDO FUNCAO  #
#":PRINT:BEEP:BEEP:D$=D$+WW$+".DRW"
980 Z=LI:OPEN D$ FOR OUTPUT AS #1
990 T$="":FOR F=&H9C40 TO DI
1000 T$=T$+CHR$(PEEK(F))
1010 LE=LEN(T$)
1020 IF LE<200 THEN NEXT F ELSE GOSUB 12
60
1030 PRINT #1,STR$(Z)+" DRAW "+CHR$(34)+
T$+CHR$(34)
1040 Z=Z+10:T$=""
1050 IF F<DI THEN NEXT F
1060 CLOSE
1070 WW$=N$
1080 PRINT:PRINT " TECLE RETURN"
1090 IF INKEY$ <> "" THEN 1090
1100 IF INKEY$ = "" THEN 1100
1110 FOR A=1 TO 4:KEY(A) ON:NEXT A:RETUR
N 110
1120 '-----------------------------------
1130 ' DESN DRAW
1140 '-----------------------------------
1150 A$=""
1160 XU=X1:YU=Y1
1170 XA=X1-X2:YA=Y1-Y2
1180 X$=STR$(XA):IF MID$(X$,1,1)=" " THE
N  X$="+"+MID$(X$,2,LEN(X$))
1190 Y$=STR$(YA):IF MID$(Y$,1,1)=" " THE
N  Y$="+"+MID$(Y$,2,LEN(Y$))
1200 A$=B$+"M"+X$+","+Y$
1210 DRAW"XA$;"
1220 FOR F=1 TO LEN(A$):L$=MID$(A$,F,1):
POKE DI+F,ASC(L$):NEXT:DI=DI+LEN(A$)
1230 RETURN
```

```
1240 X2=X1:Y2=Y1:RETURN
1250 '-------------------------------
1260 ' TRATAMENTO DE T$
1270 '-------------------------------
1280 FOR G=LE TO 1 STEP-1
1290 C$=MID$(T$,G,1)
1300 IF C$="M" THEN G=G-1:IF MID$(T$,G,1
)="B" THEN G=G-1
1310 IF C$="M" THEN 1330
1320 NEXT G
1330 T$=LEFT$(T$,G)
1340 F=F-(LE-G)
1350 RETURN
1360 '-------------------------------
1370 ' FAZ 'BACK-INSTR'
1380 '-------------------------------
1390 IF DI=&H9C3F THEN RETURN
1400 T$="":FOR F=DI TO &H9C40 STEP -1
1410 A$=CHR$(PEEK(F)):T$=A$+T$
1420 IF A$<>"M" THEN NEXT
1430 IF A$="M" AND CHR$(PEEK(F-1))="B" T
HEN T$="B"+T$:T=1
1440 IF F=&H9C3F THEN DI=&H9C3F:RETURN
1450 '-------------------------------
1460 ' FAZ 'BACK-DRAW'
1470 '-------------------------------
1480 VV=INSTR(T$,","):VM=INSTR(T$,"M")
1490 A$=MID$(T$,VM+1,VV-1):B$=MID$(T$,VV
+1,LEN(T$)-VV)
1500 IF LEFT$(A$,1)="+" THEN A$="-"+MID$
(A$,2,LEN(A$)) ELSE A$="+"+MID$(A$,2,LEN
(A$))
1510 IF LEFT$(B$,1)="+" THEN B$="-"+MID$
(B$,2,LEN(B$)) ELSE B$="+"+MID$(B$,2,LEN
(B$))
1520 T$=LEFT$(T$,VM)+A$+B$
1530 PSET(8*XU+16,8*YU+16)
1540 XU=XU+VAL(A$)
1550 YU=YU+VAL(B$)
1560 IF T=0 THEN COLOR 1:DRAW T$:COLOR 1
5
1570 IF T=1 THEN T$="":T=0:NEXT
1580 DI=F-1
1590 IF DI<&H9C3F THEN DI=&H9C3F
1600 GOSUB 390:RETURN
```

3 EDITOR DE SPRITES

Um SPRITE é um recurso gráfico do processador de vídeo que cria "máscaras" sobre a tela (screen 1, 2 ou 3).

Estas máscaras são definidas em dois tipos de matrizes: 8X8 e 16X16.

Este programa permite que se faça o desenho de um sprite na matriz desejada, transformando o mesmo numa sequência numérica facilitando assim sua colocação em uma linha DATA de um programa BASIC.

Você poderá gravar essas linhas e depois MERGEá-las (fazer um MERGE) para que possam ser usadas em um outro programa. Por exemplo, em um ANIMADOR DE SPRITES !!

O programa possui os seguintes comandos edição:

<Barra de espaços> - Preenche/apaga um ponto.
<Setas> - Movem o cursor.
<DEL/DELETE> - Apaga TODOS os Sprites definidos na memória.
[F1] DATA - Mostra na tela as linhas datas criadas.
[F2] GRAVA - Grava as linhas data.
[F3] SPR$ - Mostra o Sprite em seu tamanho real.
[F4] MUDA - Troca a matriz de 8X8 para 16X16 e vice-versa.
[F5] DEF$ - O sprite desenhado é definido na memória e está preparado para ser usado por [F1] e [F2].

Funcionamento do programa:

Da linha 70 até a linha 200 'prepara-se' o programa, ou seja, dá-se entrada a dados que o programa precisa processar. Por exemplo: na linha 110 definimos que o Sprite inicial (SI) estará no endereço 40000. As linhas entre 210 e 520 formam a rotina de processamento.

Da linha 530 a 990 estão colocadas as sub-rotinas do programa, ou melhor, as rotinas das teclas função.

Da linha 1000 em diante estão as sub-rotinas auxiliares.

Em uma linha geral o programa funciona da seguinte forma:

- Gera uma matriz onde serão desenhados os Sprites.
- Lê os pontos da matriz e os coloca na memória.
- Transforma o conjunto de números existentes na memória em linhas DATA apartir da linha inicial escolhida.

Acredito que agora deve haver a seguinte questão na sua cabeça:
Como gerar essa linha DATA?

Deve-se abrir um arquivo (ver linhas 580 e 780) com o dispositivo desejado, para depois preenche-lo com os dados que definem um sprite.

Este arquivo armazenará as informações no modo ASCII (Como se gravássemos um programa em ASCII), que pode ser "mergeado" ou carregado na memória com um comando LOAD.

Agora que você já sabe manusear o programa dê asas a imaginação e quem sabe, fazer até um desenho animado!!

```
10 '        +----------------------------+
20 '        !Aldo Barduco Junior - 88!
30 '        !     Editora Aleph          !
40 '        !   Editor de Sprites        !
50 '        +----------------------------+
60 '
70 CLEAR 1000,39999!:WIDTH(38):KEY OFF:C
OLOR 15,1,1:CLS
80 DEFSNG A-Z:MAXFILES=3
90 LOCATE 10,0:PRINT"EDITOR DE SPRITES":
PRINT :INPUT"LINHA INICIAL DOS 'DATAS'";
LI:IF LI<0 OR LI>60000! THEN 90
100 PRINT :INPUT"NOME DO ARQUIVO DE 'DAT
AS'";N$:IF LEFT$(N$,1)<"A" THEN LOCATE 0
,3:GOTO 100
110 SI=40000!
120 '-----------------------------------
130 ' ATIVA AS TECLAS DE FUNCAO
140 '-----------------------------------
150 FOR F=1 TO 5:KEY(F) ON:NEXT
160 '-----------------------------------
170 'INICIALIZA TELA
180 '-----------------------------------
190 OPEN "GRP:" FOR OUTPUT AS #1
200 SCREEN2
210 'CLS
220  LINE (0,0)-(255,191),1,BF
230  PSET(0,184),12: PRINT #1," DATA  G
```

```
RAVA SPR$ MUDA DEF$"
240 ' LE SPRITE
250 RESTORE 1220:A$="":FOR F=1 TO 8:READ
 A:A$=A$+CHR$(A):NEXT F:SPRITE$(1)=A$
260 COLOR 12:PSET(60,0),12:PRINT #1,"EDI
TOR DE SPRITES"
270  GOSUB 380
280 COLOR 15
290 ON KEY GOSUB 540,690,820,910,990
300 PUT SPRITE 1,(X,Y),11,1
310 K$=INKEY$
320 IF K$=CHR$(127) THEN SI=40000!
330 J=STICK(0)ORSTICK(1)ORSTICK(2):IF J/
2=INT(J/2) THEN J=0 ELSE J=J/2+.5
340 ON J GOSUB 510,470,490,450
350 IF STRIG(0)ORSTRIG(1)ORSTRIG(2)=-1 T
HEN GOSUB 430
360 GOTO 300
370 '-----------------------------------
380 ' STATUS DO SPRITE
390 '-----------------------------------
400 IF S=0 THEN S$="8/8" ELSE S$="16/16"
410 A=S+1:ON A GOSUB 1140,1180
420 COLOR 15:RETURN
430 BEEP:CP=POINT(X,Y):IF CP=1 THEN CP=1
5 ELSE CP=1
440 LINE(X,Y)-(X+7,Y+7),CP,BF:FOR F=1 TO
 50:NEXT:RETURN
450 IF X>IX THEN X=X-10
460 GOTO 520
470 IF X<FX THEN X=X+10
480 GOTO 520
490 IF Y<FY THEN Y=Y+10
500 GOTO 520
510 IF Y>IY THEN Y=Y-10
520 FOR F=1 TO 50:NEXT:RETURN
530 '-----------------------------------
540 ' [F1]-DATA
550 '-----------------------------------
560 CLOSE:SCREEN 0:LOCATE 10,0:PRINT"EDI
TOR DE SPRITES":PRINT"[F1]-LINHAS DATA
      LI=";LI:PRINT"-------------------
--------------------"
570 ' GERADATA 2.1
580 OPEN "CRT:DATAS" FOR OUTPUT AS #2
590 L=LI:G=40000!:IF SI=G THEN CLOSE:PRI
NT "ERRO- NENHUM SPRITE DEFINIDO.":FOR F
=1 TO 100:BEEP:NEXT F:RETURN 150
```

```
600 I=PEEK(G):I=-8*(I=56)-32*(I=49):A$="
"
610 FORF=1 TO I:A$=A$+STR$(PEEK(G+F))+",
":NEXT F:A$=STR$(L)+" DATA "+LEFT$(A$,LE
N(A$)-1):L=L+10
620 PRINT #2,A$:G=G+1+I:IF G<SI THEN 600
630 CLOSE:PRINT
640 PRINT "TECLE RETURN"
650 IF INKEY$<>"" THEN 650
660 IF INKEY$="" THEN 660
670 RETURN 150
680 '-----------------------------------
690 ' [F2]-GRAVA EM DISCO/FITA
700 '-----------------------------------
710 CLOSE:SCREEN 0:CLS
720 LOCATE 10,0:PRINT"EDITOR DE SPRITES"
:PRINT:PRINT"(1)-GRAVA EM DISCO":PRINT"(
2)-GRAVA EM FITA":PRINT"(3)-VOLTA PARA E
DICAO"
730 A$=INKEY$
740 IF A$>"3" OR A$<"1" THEN 730
750 IF A$="1" THEN B$="A:"+N$+".SPR"
760 IF A$="2" THEN B$="CAS:"+N$+".SPR"
770 IF A$="3" THEN RETURN 150
780 PRINT:PRINT"## EXECUTANDO FUNCAO ##"
:PRINT:OPEN B$ FOR OUTPUT AS #2
790 GOSUB 590
800 RETURN 150
810 '-----------------------------------
820 ' [F3]-SPRITE$ (MOSTRA)
830 '-----------------------------------
840  GOSUB 1030
850  SPRITE$(0)=B$
860 BEEP:FORF=1TO195:PUT SPRITE0,(32,F),
15,0:NEXT:BEEP:FORF=195TO100STEP-1:PUT S
PRITE0,(32,F),15,0:NEXT
870 IF INKEY$<>""THEN 870
880 IF INKEY$=""THEN 880
890 SPRITE$(0)="":RETURN
900 '-----------------------------------
910 ' [F4]-MUDA DE 8X8 PARA 16X16
920 '-----------------------------------
930 BEEP:CLOSE:S=S-(S=0)+(S=1)
940 IF S=0 THEN SCREEN,0 ELSE SCREEN,2
950 RETURN 150
960 '-----------------------------------
970 '[F5]-DEF$ (DEFINE/MOSTRA)
980 '-----------------------------------
```

```
990 GOSUB 1040:GOSUB 850:RETURN
1000 '-----------------------------------
1010 'DEFINE SPRITE NA MEMORIA
1020 '-----------------------------------
1030 DF=1
1040 C$="":B$="":POKE SI,ASC(LEFT$(S$,1)
):BEEP:FOR G=IY TO FY STEP 10
1050 A$="":FOR F=IX TO FX STEP 10:CP=POI
NT(F,G):IF CP=15 THEN A$=A$+"1"ELSE A$=A
$+"0"
1060 NEXT
1070 IF LEN (A$)>8 THEN C$=C$+CHR$(VAL("
&B"+RIGHT$(A$,8))):A$=LEFT$(A$,8)
1080 B$=B$+CHR$(VAL("&B"+A$)):NEXT:B$=B$
+C$
1090 FOR F=1 TO LEN(B$):POKE SI+F,ASC(MI
D$(B$,F,1)):NEXT F:IF DF=0 THEN SI=SI+F
ELSE DF=0
1100 RETURN
1110 '-----------------------------------
1120 'FAZ GRADE 8X8
1130 '-----------------------------------
1140 FORF=40TO120STEP10:LINE(90,F)-(170,
F),15:LINE(F+50,40)-(F+50,120),15:NEXT:X
=91:Y=41:IX=X:IY=Y:FY=Y+70:FX=X+70:PRESE
T(87,32):COLOR 13:PRINT #1,"SPRITE: "+S$
:RETURN
1150 '-----------------------------------
1160 'FAZ GRADE 16X16
1170 '-----------------------------------
1180 FORI=15TO175STEP10:LINE(80,I)-(240,
I),15:LINE(I+65,15)-(I+65,175),15:NEXT:X
=81:Y=16:IX=X:IY=Y:FX=X+150:FY=Y+150:PRE
SET (10,32):COLOR 9:PRINT #1,"SPRITE:":P
RESET (15,40):PRINT #1,S$:RETURN
1190 '-----------------------------------
1200 'SPRITE-CURSOR
1210 '-----------------------------------
1220 DATA  28, 34, 73, 93, 73, 34, 28, 0
```

SPRITEANDO A TABELA DE CARACTERES NA SCREEN 1

4

Sob este mesmo título foi publicada uma dica no "100 DICAS PARA MSX"(pág.68) onde se lia a tabela de formação de caracteres na ROM e se transferiam os valores para a tabela de formação de SPRITES (que nos SCREENs 1,2 e 3 estão no mesmo endereço de VRAM).

Se você quiser se limitar ao uso de SCREEN 1 (muito usada para a confecção de jogos), pode usar o mesmo truque de maneira rapidíssima!

Basta lembrar que, ao chamarmos a SCREEN 1 via BASIC, a tabela de caracteres (2 KB) é transferida para a VRAM a partir do endereço 0. Simultaneamente, a região de 14336 a 16383 da VRAM, cujo endereço inicial é apontado pela função BASE(9), é reservada para a tabela de formação de SPRITES.

Se alterarmos o valor de BASE(9) para 0, o VDP buscará a forma dos SPRITES exatamente no lugar onde se formaram os caracteres.

Rode o programa da figura 4.1 e digite algumas teclas para ver o efeito. Lembre-se que tudo o que foi dito até aqui está apenas na linha 100. O resto é mera demonstração de efeito.

Figura 4.1

```
100 SCREEN 1,1:BASE(9)=0:SCREEN1
110 INPUT A$
120 FOR I=1 TO LEN (A$)
130 FOR K=350 TO 100 STEP-6
140 PUTSPRITE I,(K,17*I),15,ASC(MID$(A$,I
,1))
150 NEXT K
160 NEXT I
170 B$=INPUT$(1)
180 GOTO 100
```

5 SPRITEANDO A TABELA DE CARACTERES NA SCREEN 2

Como já citamos na dica anterior, a tabela de formação de caracteres do micro pode ser carregada na área da VRAM reservada à tabela de formação dos SPRITES na região de 14336 a 16383. Em BASIC, porém, isso implica na execução de 2048 PEEKs e VPOKEs, o que torna essa transferência lenta.

Será que não existe algum truque análogo ao que usamos na SCREEN 1?

Existe e é extremamente simples: basta lembrar que na SCREEN 0 o BASE(2) indica onde o micro deve carregar a tabela de formação dos caracteres (o valor "default" do BASE(2) é 2048). Em contrapartida o BASE (14) indica, na SCREEN 2, onde começa a tabela de formação dos SPRITES (como já vimos, o "default" deste BASE é 14336).

Comandando:

```
BASE(2)=BASE(14)
```

fazemos com que, ao ser chamada a SCREEN 0, o micro carregue a tabela de caracteres a partir do endereço 14336.

Quando, a seguir, chamamos a SCREEN 2, essa região não é apagada e ela passa a usar a tabela de caracteres da antiga SCREEN 0 como tabela de sprites.

O programa da figura 5.1 dá um exemplo de aplicação dessa dica.

Note que a parte essencial do programa está apenas nas linhas 110 e 120, sendo o resto apenas uma possível demonstração do efeito (tente bolar suas próprias aplicações).

Uma sub-dica interessante está na linha 190, onde se associa uma altura espacial a uma altura musical.

Faça as devidas alterações (lembrando que nunca podemos colocar mais que 4 sprites na mesma horizontal) para bolar telas de abertura de programas ou até de vídeos.

Figura 5.1

```
100 SCREEN2,1:PLAY"s0m5000"
110 BASE(2)=BASE(14)
120 SCREEN 0:SCREEN 2
```

```
130 A$="MSX"
140 FOR K=1 TO LEN(A$)
150 B$=MID$(A$,K,1)
160 FOR X=0 TO 62.8*K
170 Y=100+50*SIN(X/10)
180 PUT SPRITE K,(X,Y),15,ASC(B$)
190 PLAY "n"+STR$(Y\2)
200 IF PLAY(0) THEN 200
210 NEXT X
220 PLAY"","c8","d8"
230 CIRCLE (X+4,Y+8),20
240 NEXT K
250 LINE (40,70)-(220,135),,B
260 PAINT (42,72)
270 GOTO 270
```

USOS PARA A COR TRANSPARENTE

O VDP do MSX gera 16 cores diferentes, que dependendo da SCREEN em uso, podem estar, gerando as cores de: borda, fundo, letras, gráficos e sprites.

Não se sabe muito bem porque, mas quando a cor transparente é usada na SCREEN 2, os comandos gráficos usados funcionam mais rápido do que se a cor de fundo fosse outra que não a transparente. Veja no programa a seguir essa comparação de tempo.

Figura 6.1

```
10 SCREEN 2:DEFINT A-Z
20 LINE (0,0)-(256,96),4,BF
30 LINE (0,96)-(256,192),5,BF
40 OPEN"grp:" AS #1
50 FOR C=0 TO 11
60 TIME =0
70 FOR L=0 TO 96
80 LINE (0,0)-(L*2,L),C,BF
90 NEXT L
100 PSET(0,97+C*8),15
110 PRINT#1,"color="C"time="TIME
120 NEXT C
130 GOTO 130
```

Ao rodar o programa você deve ter visto a diferença de tempo registrada pela variável TIME (do BASIC).

Considerando que o TIME é incrementado a cada 1/60 avos de segundo, essa diferença não é substancialmente significativa, mas é um fato curioso que acontece no micro. Usando outros comandos, a diferença pode se acentuar!

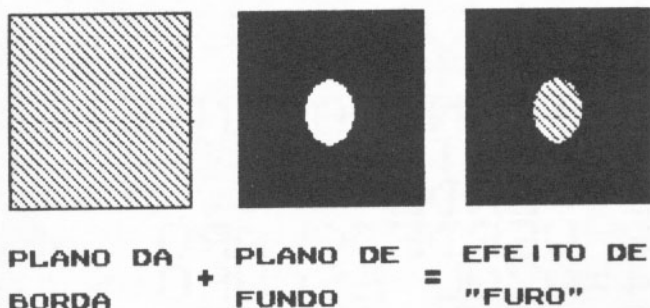
Podemos usar a cor transparente quando queremos que a confecção de uma figura complexa e demorada não seja vista quando o programa é executado.

Para que tal aconteça, basta comandar:

COLOR frente,fundo,borda

antes do comando SCREEN 2 e fazendo com que a cor de fundo seja igual a cor da borda.

A borda que você vê ao redor da SCREEN 2 é um plano que está atras do plano no qual são feitos os desenhos (plano de fundo). Se você desenhar algo com a cor transparente, ele será como um "furo" no plano dos desenhos, ou seja, o que aparecerá será o plano da borda. Observe a figura 6.2

Figura 6.2

Tendo a cor de fundo igual a cor do plano da borda, tudo o que for desenhado em cor transparente não aparecerá (já que as cores são as mesmas).

A vantagem é que você pode ter desenhos ou texto em outra cor para ser visto pelo usuário enquanto a sua figura complexa é desenhada.

Quando o desenho estiver pronto, basta mudar a cor do plano da borda para uma diferente da cor do plano de fundo comandando:

COLOR ,,nova cor para borda

Veja na figura 6.3 o efeito gerado.

Figura 6.3

```
100 OPEN"grp:" AS#1
110 COLOR 15,4,4:SCREEN 2:PSET(0,0)
120 PRINT#1,"Prepare-se para a surpresa!!"
130 LINE (0,10)-(256,10),0
140 FOR X=10TO 250 STEP 15
150 PSET (X,10)
```

```
160 FOR Y=10 TO 250 STEP X/10
170 X1=X-Y/5-5
180 IF X1<0 THEN GOTO 210
190 LINE -(X1,Y),0
200 LINE -(X-Y/2,Y+30),0
210 NEXT Y,X
220 FOR X=10 TO 250 STEP 30
230 PAINT(X,12),0:NEXT
240 COLOR ,,15
250 SOUND7,7:SOUND8,16:SOUND12,32
260 SOUND6,0:SOUND13,5:SOUND13,0
270 GOTO 270
```

```
 0=TRANSPARENTE
 1=PRETO
 2=VERDE
 3=VERDE CLARO
 4=AZUL ESCURO
 5=AZUL CLARO
 6=VERMELHO ESCURO
 7=CIANO
 8=VERMELHO
 9=VERMELHO CLARO
10=OURO
11=AMARELO
12=VERDE MUSGO
13=MAGENTA
14=CINZA
15=BRANCO
```

TITULADOR EM BASIC

7

Este programa usa os recursos gráficos do seu MSX para escrever uma mensagem em letras ampliadas, coloridas e em itálico, resultando num efeito estético muito bom. Pode ser usado para titular filmes em videocassete ou simplesmente para demonstrar a capacidade do micro.

A mensagem a ser impressa deve ser colocada em A$ e o máximo número de letras dependerá do tamanho definido nas variaveis AY e AX (altura e largura). O grau de inclinação é definido na variável IT (se valer 0, não há efeito de itálico). As cores estão definidas na linha DATA, correspondendo cada número a um oitavo do caractere.

```
10 AX=3:AY=4:IT=1
20 OPEN "GRP:" FOR OUTPUT AS 1
30 A$="UMA ROSA,SENAO SE CHA-MASSE ROSA,
SERIA MENOSBELA?"
40 SCREEN 2:X=0:Y=0:IT=1
50 FOR I=1 TO LEN(A$)
60 A=ASC(MID$(A$,I)):B=&H1BBF+8*A
70 RESTORE 210
80 E=8*IT+(6+X*7)*AX:IFE>250THENX=0:Y=Y+8
90 FOR J=0 TO 7
100 C=PEEK(B)
110 READ D:COLOR D
120 FOR K=0 TO 5
130 IF C<128 THEN 170
140 E=8*IT+K*AX+X*7*AX-J*IT:F=(Y+J)*AY
150 PSET (E,F),POINT (E,F)
160 PRINT #1,"O":PAINT (E+3,F+3)
170 C=(C*2) AND 255
180 NEXT K
190 B=B+1:NEXT J:X=X+1:NEXT I
200 GOTO 200
210 DATA 3,3,3,10,10,7,7,7
```

O programa é bem versátil, só falta fazer a divisão silábica automaticamente... . Experimente mudar as variáveis AX, AY e IT, bem como os números da linha DATA para obter efeitos diferentes.

PÁGINAS ADICIONAIS NA SCREEN 0

Se você observar o mapa de ocupação da VRAM na SCREEN 0 (veja APROFUNDANDO-SE NO MSX pág 105) verá que apenas uma pequena parcela estará ocupada. A rigor temos 960 bytes (de 0 a 959) apontados pelo BASE(0) que armazenam a tabela de nomes dos caracteres (40 colunas X 24 linhas =960 posições) e 2048 bytes são copiados a partir da ROM que armazenam o padrão de formação dos caracteres (256 caracteres X 8 bytes por caracter =2048 bytes).

Esses 2 Kbytes localizam-se normalmente entre os endereços 2048 e 4095 da VRAM. Esta tabela é apontada pelo BASE(2).

De 4096 até 16383 temos 12 Kbytes livres que podem ser usados para armazenar outras 12 telas (em cada Kbyte cabem, com folga, os 960 bytes de formação de uma SCREEN 0).

O programa "TERROMOTO" da figura 8.1 copia a tela que se forma de 0 a 959, para os endereços de 4096 em diante, deslocada de um byte.

Figura 8.1

```
100 'PROGRAMA TERREMOTO
110 SCREEN 0:WIDTH 40:KEY ON:DEFINT I
120 PRINT"ESTA MENSAGEM VAI TREMER DEVIDO
 AO DES-
130 PRINT"LOCAMENTO DE 1 BYTE NA NOVA TAB
ELA
140 PRINT"DETERMINADA PELO BASE(0)
150 PRINT"... AGUARDE 5 SEGUNDOS
160 FOR I= 0 TO 960
170 VPOKE I+4096+1,VPEEK(I)
180 NEXT I
190 FOR K=0 TO 1000
200 BASE(0)=4096
210 BASE(0)=0
220 NEXT K
230 LIST
```

Depois de 5 segundos (tempo necessário para cópia) o loop de 190 a 220 fica alternando o valor do BASE(0) de 0 (posição do original) a 4096 (posição da cópia).

Como a cópia está deslocada de 1 byte, a imagem na tela vai começar a tremer na horizontal, simulando um "terremoto" no seu vídeo.

Se você alterar a linha 170 para:

```
170 VPOKE I+4096+40,VPEEK(I)
```

você produzirá um deslocamento de 40 bytes (uma linha inteira), fazendo a tela tremer na vertical!

Mas, como já vimos, há a possibilidade de se usar 12 telas alternativas na SCREEN 0. Digite o programa da fig8.2 e veja como fazer isso.

Figura 8.2

```
100 FOR I=4096 TO 16383
110 VPOKE I,I\1024+61
115 LOCATE 0,20:PRINTI
120 NEXT I
130 FOR B=4 TO 15
140 BASE(0)=1024*B
145 FOR T=0 TO 900:NEXT T
150 NEXT B
160 GOTO 130
```

O programa enche a primeira tela alternativa com A, a segunda com B e assim sucessivamente até L. Enquanto as telas vão sendo preenchidas aparece um contador de bytes escritos que começa em 4096 e termina em 16383.

Ao terminar o preenchimento das 12 telas, elas vão sendo mostradas em seqüencia com a mudança do BASE (0) (linha 140).

Se você brecar o programa, aparentemente perderá o controle do micro, pois tudo que você digitar será mandado para a tela original (que começa em 0) e que não é mostrada no vídeo.

Não se preocupe: basta digitar (às cegas) SCREEN 0 (e RETURN) e você verá novamente o cursor. Porém, se você comandar:

```
PRINT BASE(0)
```

você obterá um valor diferente do "default" que é zero.

Quando o comando SCREEN 0 foi executado, o valor do BASE(0) era outro, então, o micro inicializou outra parte da VRAM para a tela. Mas como você pode notar, o

funcionamento é o mesmo!

É exatamente esse truque que a dica número 1 faz, mas o resto da VRAM está ocupado com tabelas para a SCREEN 2. Usando apenas a SCREEN 0, você pode mudar o valor do BASE(0) e em seguida usar o comando SCREEN 0 de modo que você possa montar telas de "HELP" de programas simplesmente usando o comando PRINT. E após essa tarefa, voltar o BASE(0) novamente a zero (tela "default"). Veja o programa exemplo da figura 8.2.

Figura 8.3

```
10 FOR I=0 TO 11
20 BASE(0)=I*1024+4096
30 SCREEN 0
40 FOR L= 1 TO 300:PRINTI;:NEXT L
60 LOCATE 0,I:PRINT"*********"
70 PRINT"*Tela";I"*":PRINT"*********"
90 NEXT I
100 FOR B=0 TO 15
110 BASE(0)=1024*B
120 FOR T=0 TO 400:NEXT T
130 NEXT B: GOTO 100
```

O loop da linha 10 a 90 muda o BASE(0) para cada uma das telas, comanda um SCREEN 0 para que possamos usar o comando PRINT naquela área da VRAM. Os comandos PRINT, preenchem cada "tela alternativa" com o seu número.

O loop da linha 100 a 130 apenas mostra cada uma das possíveis telas, até mesmo sobre a tabela de formação dos caracteres (onde há espaço para duas telas), mas que não podem ser alteradas, pois senão seria perdido o formato dos caracteres.

Note que, uma vez preenchida a VRAM, ela pode ser copiada em disco com o comando BSAVE (opção ",S"). Isto significa que você pode carregar até 12 telas de HELP com uma velocidade muito maior do que se elas fossem geradas com o comando PRINT.

Com essa dica, você pode, por exemplo, elaborar um software didático em que o texto explicativo está nas 12 telas alternativas enquanto o programa que roda na tela "default" se encarrega de apresentar testes. A qualquer momento, o usuário-aluno pode consultar a teoria (com F1) e depois voltar ao teste (com F2) para responder corretamente.

Este é apenas um exemplo de aplicação: existem "trocentos" outros! Mãos à obra!

BRINCANDO COM O Y DOS SPRITES

Quando você define e localiza SPRITEs nas SCREENs 1, 2 ou 3, existe uma tabela de 128 bytes, denominada TABELA DE ATRIBUTOS DOS SPRITES localizada a partir do endereço &H1B00 da VRAM. Este endereço é setado pelo BASE conforme a tabela a seguir:

SCREEN 1 - BASE(8)
SCREEN 2 - BASE(13)
SCREEN 3 - BASE(18)

Esses 128 bytes estão divididos em 32 grupos(1 para cada camada da tela)de 4 bytes assim distribuídos:

byte 0 - coordenada y do SPRITE
byte 1 - coordenada x do SPRITE
byte 2 - código da cor do SPRITE
byte 3 - número do SPRITE

Por algum estranbo motivo, que foge totalmente à nossa compreensão, quando você coloca um sprite na posição y(ou através do PUTSPRITE ou dando um VPOKE no byte 0 do grupo adequado da tabela de atributos) este sprite não aparece na tela na posição indicada mas sim um pixel para baixo, ou seja, na posição y+1!

Digite o programa da figura 9.1 para perceber este fenômeno.

Figura 9.1

```
100 SCREEN 2,1:STRIG(0) ON
110 ON STRIG GOSUB 210
120 OPEN"GRP:" AS #1
130 DATA FF,81,BD,A5,A5,BD,81,FF
140 FOR F=1 TO 8
150   READ A$:A=VAL("&H"+A$)
160   S$=S$+CHR$(A)
170 NEXT F
180 SPRITE$(0)=S$
190 LINE(80,80)-(95,95),11,BF
200 GOTO 200
210 STRIG(0) OFF:N=N+1:C=N MOD 2
220 PUT SPRITE 0,(80,80-C),6*(C+1),0
```

```
230 LINE (80,40)-(200,78),1,BF
240 FOR B=0 TO 3
250 PRESET (80,40+10*B)
260 PRINT#1,"BYTE";B;"->";
270 PRINT#1,VPEEK(&H1B00+B)
280 NEXT B
290 STRIG(0) ON:RETURN
```

Ao rodar o programa, a linha 190 desenha um quadrado cujo vértice superior esquerdo está nas coordenadas(80,80). Ao pressionar a barra de espaços, a interrupção habilitada na linha 100 desvia o programa para a sub-rotina 210 e o sprite definido no laço 140-170 é colocado exatamente em cima deste quadrado.

O laço 240-280 lê os 4 bytes do primeiro grupo (camada 0 na qual estamos trabalhando).

Note que, enquanto o byte 1 (do x) contém o valor correto (80), o byte 0 (do y) apresenta o valor 79, sem o qual não haveria coincidência.

Apertando novamente a barra de espaços, o valor de C (linha 210) é atualizado (ele fica se alternando entre 0 e 1) e o valor do byte 0 é corrigido para 80. O sprite, nesse momento, deixa de coincidir com o quadrado. Aperte a barra de espaços algumas vezes para se convencer e levar isso em consideração a próxima vez que tiver que localizar com precisão um sprite na tela.

Com relação a este programinha vale a pena analisar como habilitar e desabilitar uma interrupção (linha 100 e 210), como formar um sprite com linha DATA (laço 140-170), como gerar um número que assuma,alternadamente os valores 1 e 0 (linha 210), como escrever na tela gráfica (linhas 120,260 e 270) e como apagar nesta tela (linha 230).

Uma outra particularidade deste tal byte 0 do grupo de 4 byte de uma camada, refere-se ao código &HD0 (208 em decimal).

Se você escrever este valor no byte 0 de alguma camada, não só esse sprite vai desaparecer (o que é de se esperar: você o está colocando além do 191, ou seja, fora da tela) mas vão desaparecer todos os outros sprites situados em camadas de número superior.

É um bom truque para fazer sumir de maneira instantânea uma grande quantidade de sprites!

Digite o programa da figura 9.2 e rode-o. Não se assuste com seu tamanho: as linhas de 110 a 590 apenas definem 10 sprites (16x16) muito bonitos que você

poderá usar em outros programas (em associação com o "carimbador de sprites" do CEM DICAS PARA MSX permitem formar lindas texturas de fundo de tela).

Figura 9.2

```
100 ' 0
110 DATA 28,2B,C0,0D,D0,13,44,55
120 DATA 55,44,13,D0,0D,C0,2B,28
130 DATA 14,D4,03,B0,0B,C8,22,AA
140 DATA AA,22,C8,0B,B0,03,D4,14
150 ' 1
160 DATA FF,C0,BF,B0,AF,AC,AB,AA
170 DATA AA,AB,AC,AF,B0,BF,C0,FF
180 DATA FF,03,FD,0D,F5,35,D5,55
190 DATA 55,D5,35,F5,0D,FD,03,FF
200 ' 2
210 DATA 49,89,12,22,C4,09,31,C6
220 DATA C6,31,09,C4,22,12,89,49
230 DATA 92,91,48,44,23,90,8C,63
240 DATA 63,8C,90,23,44,48,91,92
250 ' 3
260 DATA FF,3C,42,99,A5,A5,A5,A5
270 DATA 9F,A0,CF,D0,D0,CF,A0,9F
280 DATA F9,05,F3,0B,0B,F3,05,F9
290 DATA A5,A5,A5,A5,99,42,3C,FF
300 ' 4
310 DATA E0,96,96,70,0E,69,69,07
320 DATA F8,96,96,F1,8F,69,69,1F
330 DATA F8,96,96,F1,8F,69,69,1F
340 DATA E0,96,96,70,0E,69,69,07
350 ' 5
360 DATA A6,29,EA,0A,7A,82,BE,41
370 DATA 41,BE,82,7A,0A,EA,29,A6
380 DATA 65,94,57,50,5E,41,7D,82
390 DATA 82,7D,41,5E,50,57,94,65
400 ' 6
410 DATA FF,81,BD,A5,A5,BD,81,FF
420 DATA FF,81,BD,A5,A5,BD,81,FF
430 DATA FF,81,BD,A5,A5,BD,81,FF
440 DATA FF,81,BD,A5,A5,BD,81,FF
450 ' 7
460 DATA A5,5A,A5,5A,5A,A5,5A,A5
470 DATA A5,5A,A5,5A,5A,A5,5A,A5
480 DATA A5,5A,A5,5A,5A,A5,5A,A5
490 DATA A5,5A,A5,5A,5A,A5,5A,A5
500 ' 8
510 DATA AA,D5,6A,B5,5A,AD,56,AB
```

```
520 DATA AB,56,AD,5A,B5,6A,D5,AA
530 DATA 55,AB,56,AD,5A,B5,6A,D5
540 DATA D5,6A,B5,5A,AD,56,AB,55
550 ' 9
560 DATA FF,80,BF,A0,AF,A8,AB,AA
570 DATA AA,AA,AA,AB,A8,AF,A0,BF
580 DATA FF,01,FD,05,F5,15,D5,55
590 DATA D5,D5,15,F5,05,FD,01,FF
600 '
610 DEFINT A-Z
620 SCREEN 2,2:OPEN"GRP:" AS#1
630 FOR F=0 TO 9
640   S$=""
650   FOR G=1 TO 32
660     READ A$:A=VAL("&H"+A$)
670     S$=S$+CHR$(A)
680   NEXT G
690   SPRITE$(F)=S$
700 NEXT F
710 FOR F=0 TO 9
720   PUT SPRITE F,(20*F,10*F),15,F
730   PRESET(20*F,10*F+25):PRINT#1,F
740 NEXT F
750 A$=INPUT$(1):F=VAL(A$)
760 VPOKE &H1B00+4*F,&HD0
770   FOR G=1 TO 500 : NEXT G
780 GOTO 710
```

O laço 630-700 lê as linhas DATA e forma 10 sprites (de 0 a 9). O laço 710-740 coloca os 10 sprites na tela com sua numeração.

A linha 750 espera que você digite um número (de 0 a 90).A linha 760 coloca o código &HD0 no byte 0 do grupo de atributos correspondentes.

Neste instante você verá o sprite em questão desaparecer, acompanhado de todos os que estão nas camadas de código maior.

As linhas 770 e 780 dão um tempo e fazem o programa voltar à 710 para que você digite outro número de 0 a 9.

Note que, se você digitar qualquer tecla não numérica o F=VAL A$ assume o valor 0 e todos os sprites irão desaparecer.

TABELA DE CARACTERES DO PC

10

Quando trabalhamos com o drive no MSX, temos à nossa disposição o MSX-DOS, ou algum de seus clones nacionais.

Isso nos abre um universo muito vasto pois como o MSX-DOS respeita os padrões do CP/M podemos atuar mais profissionalmente com o MSX, que é considerado por muitos como um "video-game de luxo" (isso apenas pelo fato de aceitar cartuchos em vez de "placas peladas").

Trabalhando em MXS-DOS, temos à nossa disposição muitas linguagens e programas (LISP, C, PASCAL, ASSEMBLER, COBOL, dBASE, etc) de variadas soft-houses.

Uma característica também muito interessante é a compatibilidade a nível de arquivos com o PC, pois não precisamos de nenhum programa conversor para ler os seus arquivos e vice-versa. Até mesmo a estrutura do diretório é idêntica.

Alguns programas em BASIC do PC rodam no MSX (desde que possuam apenas os comandos "standard" do BASIC).

O mesmo acontece com programas em linguagens que rodam sob o MSX-DOS.

Analisemos o caso, por exemplo, do turbo-pascal. Existe uma versão para MSX e uma versão para PC. Os dois são programas em linguagem de máquina feitos para computadores diferentes com microprocessadores diferentes. Ou seja um turbo pascal para MSX nunca vai rodar em um PC e vice versa.

Porém esses dois programas são compiladores de uma mesma linguagem (o PASCAL). Eles transformam um arquivo texto (que contęnha os comandos standard do PASCAL) em um programa em linguagem de máquina (cada um deles para o seu microprocessador especificamente).

Então, um mesmo programa fonte pode ser compilado nos dois computadores diferentes sem nenhuma dificuldade.

Porém, a tabela de caracteres do PC é diferente da do MSX. Ela apresenta uma série de caracteres gráficos (principalmente de molduras) que dão um toque muito especial aos programas.

Portanto, para que você ao pegar um desses programas consiga entender o efeito a ser gerado na tela é que nós "chupamos" parte da tabela de caracteres do PC que difere da de um MSX e a colocamos

no programada figura 10.1.

As linhas DATA contêm a matriz dos caracteres (um caracter por linha - 8 bytes) e os dois loops no final do programa redefinem a tabela de caracteres (screen 0).

Após rodar o programa, você pode gravar a tabela comandando:

```
BSAVE"CARACPC.TAB",2048,4096,S
```

e sempre que quiser recuperar:

```
BLOAD"CARACPC.TAB",S
```

Figura 10.1

```
1000 DATA 80,C0,E0,F0,E0,C0,80,00
1010 DATA 04,0C,1C,3C,1C,0C,04,00
1020 DATA 20,70,20,20,20,70,20,00
1030 DATA 50,50,50,50,50,00,50,00
1040 DATA 7C,A8,A8,68,28,28,28,00
1050 DATA 3C,40,38,44,38,04,78,00
1060 DATA 00,00,00,FF,FF,00,00,00
1070 DATA 20,70,20,20,20,70,20,FF
1080 DATA 20,70,20,20,20,20,20,00
1090 DATA 20,20,20,20,20,70,20,00
1100 DATA 00,00,08,FC,08,00,00,00
1110 DATA 00,00,40,FE,40,00,00,00
1120 DATA 00,00,00,80,80,F0,00,00
1130 DATA 00,00,44,FE,44,00,00,00
1140 DATA 00,00,00,00,30,78,FC,00
1150 DATA FC,78,30,00,00,00,00,00
1160 DATA 00,30,48,84,84,84,FC,00
1170 DATA 30,48,80,80,80,48,30,40
1180 DATA 00,90,00,90,90,90,68,00
1190 DATA 08,10,70,88,F8,80,70,00
1200 DATA 20,50,70,08,78,88,78,00
1210 DATA 50,00,70,08,78,88,78,00
1220 DATA 80,40,70,08,78,88,78,00
1230 DATA 20,50,70,08,78,88,78,00
1240 DATA 00,00,70,88,80,88,70,20
1250 DATA 20,50,70,88,F8,80,70,00
1260 DATA 50,00,70,88,F8,80,70,00
1270 DATA 80,40,70,88,F8,80,70,00
1280 DATA 50,00,60,20,20,20,70,00
1290 DATA 20,50,00,60,20,20,70,00
1300 DATA 80,40,00,60,20,20,70,00
1310 DATA 50,A0,50,88,F8,88,88,00
```

```
1320 DATA 20,20,50,88,F8,88,88,00
1330 DATA 10,20,F8,80,F0,80,F8,00
1340 DATA 00,00,6C,12,7E,90,6E,00
1350 DATA 3E,50,90,9C,F0,90,9E,00
1360 DATA 20,50,70,88,88,88,70,00
1370 DATA 50,00,70,88,88,88,70,00
1380 DATA 80,40,70,88,88,88,70,00
1390 DATA 60,90,00,90,90,90,68,00
1400 DATA 80,40,90,90,90,90,68,00
1410 DATA 90,00,90,90,B0,50,10,E0
1420 DATA 50,70,88,88,88,88,70,00
1430 DATA 50,00,88,88,88,88,70,00
1440 DATA 20,20,78,80,80,78,20,20
1450 DATA 18,24,20,F8,20,E2,5C,00
1460 DATA 88,50,20,F8,20,F8,20,00
1470 DATA 40,A0,80,9E,84,A8,5E,00
1480 DATA 18,20,20,F8,20,20,20,40
1490 DATA 10,20,70,08,78,88,78,00
1500 DATA 10,20,00,60,20,20,70,00
1510 DATA 08,10,70,88,88,88,70,00
1520 DATA 10,20,90,90,90,90,68,00
1530 DATA 50,A0,00,A0,D0,90,90,00
1540 DATA 28,50,00,C8,A8,98,88,00
1550 DATA 60,90,90,68,00,F8,00,00
1560 DATA 60,90,90,60,00,F0,00,00
1570 DATA 20,00,20,40,80,88,70,00
1580 DATA 00,00,00,F8,80,80,00,00
1590 DATA 00,00,00,F8,08,08,00,00
1600 DATA 84,88,90,A8,54,84,08,1C
1610 DATA 84,88,90,A8,58,A8,3C,08
1620 DATA 20,00,00,20,20,20,20,00
1630 DATA 00,12,24,48,90,48,24,12
1640 DATA 00,90,48,24,12,24,48,90
1650 DATA 92,49,92,49,92,49,92,49
1660 DATA AA,55,AA,55,AA,55,AA,55
1670 DATA 6D,B6,6D,B6,6D,B6,6D,B6
1680 DATA 10,10,10,10,10,10,10,10
1690 DATA 10,10,10,F0,10,10,10,10
1700 DATA 10,10,F0,10,F0,10,10,10
1710 DATA 28,28,28,E8,28,28,28,28
1720 DATA 00,00,00,F8,28,28,28,28
1730 DATA 00,00,F0,10,F0,10,10,10
1740 DATA 2A,28,E8,08,E8,28,28,28
1750 DATA 28,28,28,28,28,28,28,28
1760 DATA 00,00,F8,08,E8,28,28,28
1770 DATA 28,28,E8,08,F8,00,00,00
1780 DATA 28,28,28,F8,00,00,00,00
1790 DATA 10,10,F0,10,F0,00,00,00
```

```
1800 DATA 00,00,00,F0,10,10,10,10
1810 DATA 10,10,10,1F,00,00,00,00
1820 DATA 10,10,10,FF,00,00,00,00
1830 DATA 00,00,00,FF,10,10,10,10
1840 DATA 10,10,10,1F,10,10,10,10
1850 DATA 00,00,00,FF,00,00,00,00
1860 DATA 10,10,10,FF,10,10,10,10
1870 DATA 10,10,1F,10,1F,10,10,10
1880 DATA 28,28,28,2F,28,28,28,28
1890 DATA 28,28,2F,20,3F,00,00,00
1900 DATA 00,00,3F,20,2F,28,28,28
1910 DATA 28,28,EF,00,FF,00,00,00
1920 DATA 00,00,FF,00,EF,28,28,28
1930 DATA 28,28,2F,20,2F,28,28,28
1940 DATA 00,00,FF,00,FF,00,00,00
1950 DATA 28,28,EF,00,EF,28,28,28
1960 DATA 10,10,FF,00,FF,00,00,00
1970 DATA 28,28,28,FF,00,00,00,00
1980 DATA 00,00,FF,00,FF,10,10,10
1990 DATA 00,00,00,FF,28,28,28,28
2000 DATA 28,28,28,3F,00,00,00,00
2010 DATA 10,10,1F,10,1F,00,00,00
2020 DATA 00,00,1F,10,1F,10,10,10
2030 DATA 00,00,3F,20,2F,28,28,28
2040 DATA 28,28,28,FF,28,28,28,28
2050 DATA 10,10,FF,10,FF,10,10,10
2060 DATA 10,10,10,F0,00,00,00,00
2070 DATA 00,00,00,1F,10,10,10,10
2080 DATA FF,FF,FF,FF,FF,FF,FF,FF
2090 DATA 00,00,00,00,FF,FF,FF,FF
2100 DATA F0,F0,F0,F0,F0,F0,F0,F0
2110 DATA 0F,0F,0F,0F,0F,0F,0F,0F
2120 DATA FF,FF,FF,FF,00,00,00,00
2130 DATA 00,00,68,90,90,90,68,00
2140 DATA 30,48,48,70,48,48,70,C0
2150 DATA F8,88,80,80,80,80,80,00
2160 DATA F8,50,50,50,50,50,98,00
2170 DATA F8,88,40,20,40,88,F8,00
2180 DATA 00,00,78,90,90,90,60,00
2190 DATA 00,50,50,50,50,68,80,80
2200 DATA 00,50,A0,20,20,20,20,00
2210 DATA F8,20,70,A8,A8,70,20,F8
2220 DATA 20,50,88,F8,88,50,20,00
2230 DATA 70,88,88,88,50,50,D8,00
2240 DATA 30,40,40,20,50,50,50,20
2250 DATA 00,00,00,50.A8,A8,50,00
2260 DATA 08,70,A8,A8,A8,70,80,00
2270 DATA 38,40,80,F8,80,40,38,00
```

```
2280 DATA 70,88,88,88,88,88,88,00
2290 DATA 00,00,F0,00,F0,00,F0,00
2300 DATA 20,20,F8,20,20,00,F8,00
2310 DATA C0,30,08,30,C0,00,F8,00
2320 DATA 18,60,80,60,18,00,F8,00
2330 DATA 10,28,20,20,20,20,20,20
2340 DATA 20,20,20,20,20,20,A0,40
2350 DATA 00,20,00,F8,00,20,00,00
2360 DATA 00,50,A0,00,50,A0,00,00
2370 DATA 00,18,24,24,18,00,00,00
2380 DATA 00,30,78,78,30,00,00,00
2390 DATA 00,00,00,00,30,00,00,00
2400 DATA 3E,20,20,20,A0,60,20,00
2410 DATA A0,50,50,50,10,10,00,00
2420 DATA 40,A0,40,80,E0,00,00,00
2430 DATA 00,00,1C,1C,1C,00,00,00
2440 FOR L=0 TO 255
2450 IF L<32 THEN PRINTCHR$(1)CHR$(64+L);
:ELSE PRINTCHR$(L);
2460 NEXT
2470 FOR L=2176 TO 2303
2480 READ A$:VPOKE L,VAL("&H"+A$)
2490 NEXT
2500 FOR L=3072 TO 4095
2510 READ A$:VPOKE L,VAL("&H"+A$)
2520 NEXT
```

Após rodar o programa, a tabela de caracteres do seu MSX ficará como mostra a figura 10.2.

Use esse programa em conjunto com a dica 31, pois assim será mais fácil achar o caracter que você quer no vídeo.

Figura 10.2

CóPIA GRÁFICA DOBRADA E QUADRUPLICADA

No livro "100 DICAS PARA MSX" já publicamos um programa (em Linguagem de Máquina) que permite fazer cópias gráficas transferindo o conteúdo da tela na impressora. Por outro lado, no "APROFUNDANDO-SE NO MSX", publicamos um programa em BASIC, e portanto de mais fácil compreensão, que permitia copiar "deitada" a SCREEN 2 na impressora.

Vários usuários nos escreveram, solicitando um programa que permitisse efetuar a cópia da SCREEN 2, mas ampliada. Isso nos motivou a escrever essa dica, alterando o programa em BASIC já citado, de maneira a transformar cada "pixel" (ponto da tela) em 4 (2X2) pontos da impressora.

Digite o programa da figura 11.1 e rode-o em um MSX ligado a uma impressora que entre em modo gráfico segundo o padrão EPSON (Olívia, Mônica, Grafix MTA, etc).

Figura 11.1

```
100  '*********************************
110  'ROTINA P/FAZER DESENHO NA TELA 2
120  '*********************************
130  KEY OFF
140  CP=15:CT=8:COLOR CT,CP:SCREEN2
150  MAXFILES=2:OPEN"GRP:" AS#1
160  PRESET(6,10):PRINT#1,"TESTE"
170  PRESET(6,20):PRINT#1,"DE"
180  PRESET(6,30):PRINT#1,"IMPRESSÃO"
190  LINE(0,0)-(255,191),,B
200  FOR FI=0 TO 25.12 STEP .157
210  LINE (140,100)-(140-90*SIN(FI),100-90
*COS(FI)):LINE-(140-(80-3.575*FI)*SIN(FI+
.157),100-(80-3.575*FI)*COS(FI+.157))
220  NEXT FI
230  GOSUB 5000:END
5000  '***********************
5010  'ROTINA PARA IMPRESSAO
5020  '***********************
5030  POKE &HF417,1:CP=15:CT=8
5040  DEFINT A-Z
5050  OPEN "LPT:" AS #2
5060  LPRINT CHR$(27);CHR$(65);CHR$(8)
```

```
5070 FOR C=0 TO 31
5080 FOR K=1 TO 2
5090 LPRINT CHR$(27);CHR$(75);CHR$(128);
CHR$(1);
5100 FOR L=23 TO 0 STEP -1
5110 FOR X= 7 TO 0 STEP -1
5120 U=VPEEK ((C*8+256*L)+X)
5130 V=VPEEK((C*8+256*L)+X+8192)
5140 IF V MOD 16= CT THEN U=255
5150 IF INT(V/16)=CP THEN U=0
5160 GOSUB 6000
5170 FOR J=1 TO 2
5180 PRINT#2,CHR$(W);
5190 NEXT J
5200 NEXT X
5210 NEXT L
5220 LPRINT CHR$(10);
5230 NEXT K
5240 NEXT C
5250 LPRINT CHR$(27);CHR$(65);CHR$(13)
5260 RETURN
6000 '************************
6010 'ROTINA PARA DUPLICAR BITS
6020 '************************
6030 W1$="":W$=RIGHT$("0000000"+BIN$(U),8)
6040 FOR P=1 TO 4
6050 P$=MID$(W$,4*K-4+P,1)
6060 W1$=W1$+P$+P$
6070 NEXT P
6080 W= VAL("&B"+W1$)
6090 RETURN
```

As linhas de 100 a 230 simplesmente definem o desenho a ser feito na SCREEN 2 e podem ser alteradas a seu bel-prazer.

Apenas tome cuidado para usar os mesmos códigos de "cor de papel" (CP) e "cor de tinta" (CT) na linha 140 e na linha 5030 da rotina de impressão.

As rotinas de impressão estão entre a linha 5000 e 6090.

A linha 5030 desativa o filtro BRASCII do micro e define os códigos de "cor do papel" e "cor da tinta".

A linha 5040 assume todas as variáveis como inteiras para tornar a execução mais rápida (taí uma boa dica para você usar em outras oportunidades) e a linha 5050 abre o dispositivo IMPRESSORA (LPT:) para receber os dados com o PRINT #2 (veja linha 5180).

A linha 5060 comanda o entrelinhamento de

impressão para oito pontos, de maneira a "emendar" cada faixa de impressão na anterior (veja a linha 5250 que restabelece um entrelinhamento mais folgado).

A linha 5090 estabelece a largura da impressão. Como a cópia é deitada, a largura da impressão é a altura da tela. No programa originalmente publicado no APROFUNDANDO-SE NO MSX, onde cada ponto da tela correspondia a um ponto na impressora, essa largura correspondia a 192 (altura da SCREEN 2) e os dois últimos bytes desse comando eram 192 e 0 pois:

192+256*0=192

Neste programa queremos "dobrar" o tamanho da cópia e precisamos de 192*2=384 pontos. Os dois últimos bytes do comando citado devem valer, então, 128 e 1 pois:

128 +256*1=384

As linhas 5070, 5080 e 5100 a 5150 lêem os bytes da VRAM para que sejam enviados à impressora (explicações mis detalhadas desssa leitura estão no livro APROFUNDANDO-SE NO MSX). A linha 5080 faz com que cada linha da tela seja lida duas vezes: na primeira vez imprime-se a metade superior do byte duplicada, na segunda vez é impressa a metade inferior duplicada.

Isso garante a duplicação na vertical.

O laço de 5170 a 5190 garante a duplicação na horizontal.

A duplicação na vertical não é muito simples e é realizada pela sub-rotina que começa em 6000.

Para entendê-la melhor, imagine que você precise "duplicar" o byte representado na figura 11.2 (quadradinho "aceso" corresponde ao 1 e "apagado ao 0).

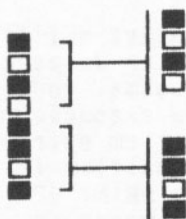
Figura 11.2

Ele é dividido em duas partes e cada uma é duplicada pela sub-rotina 6000 (figura 11.3)

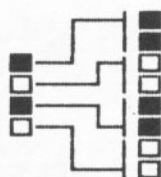
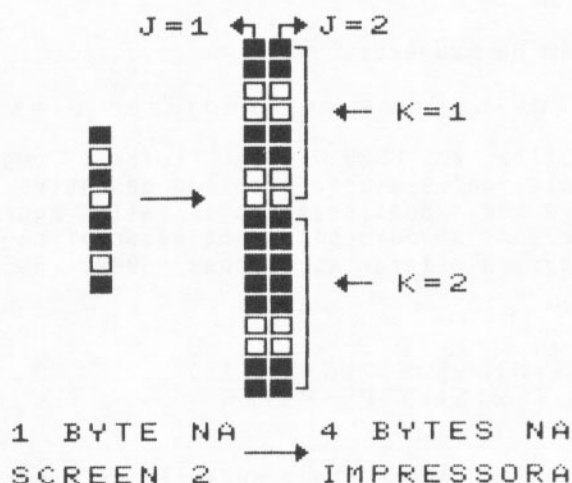
Figura 11.3

Desta forma, cada bit é duplicado na vertical e impresso duas vezes para duplicar na horizontal (figura 11.4)

Figura 11.4

Se você dispõe de uma impressora de 132 colunas, poderá até quadruplicar sua cópia gráfica.

Antes de continuar a leitura, analise o programa da figura 11.1 e veja onde devem ser introduzidas as alterações para fazer esta cópia quadruplicada.

Pronto? Vamos então checar:

Você deve ter mudado a linha 5090. Afinal a largura é o quádruplo de 192 pontos:

192*4=768

Portanto os dois últimos bytes da linha 5090 devem ser 0 e 3 pois:

0+256*3=768

ficando então:

```
5090 LPRINT CHR$(27);CHR$(75);CHR$(0);CH
R$(3);
```

A linha 5080 deve, obviamente, ser alterada para

```
5080 FOR K=1 TO 4
```

Da mesma forma, a linha 5170 mudará para:

```
5170 FOR J=1 TO 4
```

Além da mudança:

```
6010 'Rotina para quadruplicar bits
```

a subrotina em 6000 deve sofrer uma mudança importante: antes o byte era lido de quatro em quatro bits para que a duplicação fosse feita. Agora, devemos lê-lo de dois em dois bits para quadruplicá-lo. Assim sendo devemos alterar as linhas 6040, 6050 e 6060 para:

```
6040 FOR P=1 TO 2
6050 P$=MID$(W$,2*K-2+P,1)
6060 W1$=W1$+P$+P$+P$+P$
```

Fica como exercício a sugestão de alteração para triplicar a cópia. Aviso: não é simples, mas é um bom quebra cabeça. Nossa sugestão é que você divida o byte em 4 partes (4 leituras), triplique cada um dos dois bits e faça a impressora imprimir apenas seis pontos verticais em cada pasada. Obviamente o avanço de entrelinhamento da linha 5060 deverá ser alterado para seis e não mais para 8. Bom divertimento!

EDITOR DE "FAIXAS"

12

Quem já lidou com um computador compatível com Apple ou PC, com certeza já utilizou um software denominado PRINTSHOP. Entre outras coisas o PRINTSHOP permite fazer "faixas" ou seja, escrever deitado na impressora de maneira a produzir longas tiras com mensagens.

Esta dica foi escrita para dar ao usuário de MSX o recurso de editar suas próprias faixas, usando vários tamanhos de letras com várias texturas.

Você precisa ter uma impressora que entre no modo gráfico segundo o padrão EPSON (Mônica, Grafix, Olivia, etc) e que tenha 80 ou 132 colunas.

Digite o programa da figura 12.2 , tomando cuidado com erros de digitação e gravando-o periodicamente para não perder tudo em caso de problemas com a energia elétrica.

Ao rodar o programa, a primeira mensagem é:

```
QUAL A LARGURA DO CARACTER(6/7/8)?
```

Se você digitar 6, terá o espaçamento da SCREEN 0 do MSX. Obviamente alguns caracteres gráficos aparecerão cortados (como acontece na SCREEN 0 da tela do MSX).

Se você digitar 8, terá o espaçamento da SCREEN 1, aparecendo todos os caracteres gráficos, integralmente. Existe a opção intermediária do 7, que dará um espaçamento entre o da SCREEN 0 e o da SCREEN 1.

A seguir, aparecerá a mensagem:

```
QUAL A LARGURA DA IMPRESSAO(1 A 32)?
```

Como estamos fazendo uma faixa, com caracteres deitados, a largura da impressão é medida na vertical; cada unidade equivale a 8 agulhas da impressora. Isto significa que, se você escolher 1, seu pixel na faixa terá uma largura correspondente à largura de um caracter da impressora.

A mensagem:

```
QUAL A ALTURA DA IMPRESSAO(1 A 14)?
```

determina a altura do pixel (medida na horizontal).

O máximo valor para uma impressora de 80 colunas é 9 e, para uma de 132 colunas é 14.

Se você quiser um pixel quadrado, deverá usar o mesmo valor para a largura e a altura de impressão.

Usando valores diferentes, as letras e caracteres gráficos serão achatados ou esticados.

A seguir, aparece um menu de opções para a textura do traço. Você pode escolher desde a opção 1 (traço fino acionando uma agulha sim e outra não alternadamente) até a 4 (traço cheio ativando todas as 8 agulhas).

Na figura de 12.1 você tem alguns exemplos, em tamanho natural, destas combinações.

Figura 12.1

Finalmente o programa solicita a mensagem a ser transformada em faixa.

Não digite mensagens muito longas por 2 motivos: o programa só aceita strings de, no máximo, 250 caracteres e a impressão é lenta. Se você exagerar no tamanho, pode demorar um bocado de tempo até obter sua faixa.

Cuidado ao utilizar caracteres gráficos de código inferior a 32: eles contam em dobro no tamanho da string!

Figura 12.2

```
100 CLEAR1024:POKE&HF417,255:DEFINT A-Z
110 LPRINTCHR$(27)"A"CHR$(8);
120 SCREEN 0:WIDTH 40
130 INPUT"QUAL A LARGURA DO CARACTER(6/7
/8)";LA
```

```
140 LA=INT(LA)
150 IF LA<6 OR LA>8 THEN 130
160 INPUT"QUAL A LARGURA DA IMPRESSAO(1
A 32)";LI
170 LI=INT(LI)
180 IF LI<1 OR LI>32 THEN 160
190 INPUT"QUAL A ALTURA DA IMPRESSAO(1 A
 14)";HI
200 HI=INT(HI)
210 IF HI<1 OR HI>14 THEN 190
220 HP=HI*64:MSB=HP\256:LSB=HP MOD 256
230 PRINT "TRACO FINO    - 1"
240 PRINT "TRACO MEDIO   - 2"
250 PRINT "TRACO GROSSO - 3"
260 PRINT "TRACO CHEIO   - 4"
270 WW$=INPUT$(1):WW=INT(VAL(WW$))
280 IF WW<1 OR WW>4 THEN 230
290 IF WW=1 THEN WW=170
300 IF WW=2 THEN WW=204
310 IF WW=3 THEN WW=240
320 IF WW=4 THEN WW=255
330 SCREEN 1
340 INPUT"QUAL A MENSAGEM";A$
350 IF LEN(A$)>250 THEN CLS:PRINT"LONGA
DEMAIS!":GOTO 340
360 GOSUB 740
370 FOR I=1 TO LEN(A$)
380  PRINT
390  C$=MID$(A$,I,1)
400  E=ASC(C$)*8
410  FOR K=0 TO 7
420   X=E+K
430   Y$=BIN$(VPEEK(X))
440   B$(K)=RIGHT$("00000000"+Y$,8)
450   PRINTB$(K)
460  NEXT K
470  PRINT
480  FOR F=1 TO LA
490   P$=""
500   FOR J=7 TO 0 STEP-1
510    P$=P$+MID$(B$(J),F,1)
520   NEXT J
530   PRINTP$:GOSUB 580
540  NEXT F
550 NEXT I
560 LPRINTCHR$(27)"A"CHR$(12)
570 END
580 REM ROTINA DE IMPRESSAO
```

```
590 FOR M=1 TO LEN(P$)
600  T$=MID$(P$,M,1)
610  T(M)=VAL(T$)*WW
620 NEXT M
630 FOR N=1 TO LI
640  IF VAL("&B"+P$)=0 THEN 710
650  LPRINT CHR$(27)"K"CHR$(LSB)CHR$(MSB
);
660  FOR M=1 TO LEN(P$)
670   FOR G=1 TO 8*HI
680    LPRINT CHR$(T(M));
690   NEXT G
700  NEXT M
710  LPRINT CHR$(10);
720 NEXT N
730 RETURN
740 REM ROTINA PARA LIMPAR A$
750 Q$=""
760 FOR R=1 TO LEN(A$)
770  S$=MID$(A$,R,1):S=ASC(S$)
780  IF S>31 THEN Q$=Q$+S$
790  IF S=1 THEN Q$=Q$+CHR$(ASC(MID$(A$,
R+1,1))-64):R=R+1
800 NEXT R
810 A$=Q$
820 RETURN
```

Vamos agora entender a estrutura do programa. Se você estudar este trecho atentamente, não só entenderá uma série de truques úteis (cada um valendo uma dica!) mas também terá condições de alterar o programa a seu gosto de maneira a inserir outros recursos de que sinta necessidade!

100 reserva 1k de memória para as strings e desativa o filtro BRASCII do MSX.

110 delimita o avanço do papel para o equivalente a 8 agulhas (o espaçamento normal da impressora é maior que isso para que sobre espaço entre uma linha e outra).

120-360 solicita a entrada dos parâmetros de impressão:

LA= largura do caracter
LI= largura do pixel a ser impresso
HI= altura do pixel a ser impresso
WW= tipo de textura
A$= mensagem a ser impressa

Para se colocar uma impressora EPSON no modo grafico, devemos enviar a seguinte sequência de caracteres:

```
LPRINT CHR$(27)"K"CHR$(LSB)CHR$(MSB);
```

que é o que faz a linha 650. Os dois últimos bytes definem a largura da impressão segundo a equação:

LARGURA GRAFICA= LSB+256*MSB

A linha 220 calcula o valor desses dois bytes em função da altura escolhida para o pixel.

Na linha final (360) do trecho que estamos examinando, o programa desvia para a subrotina em 740: ROTINA PARA LIMPAR A$.

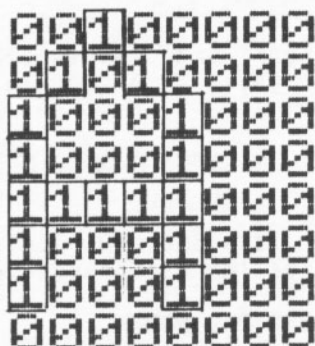
740-820: por que "limpar" A$? Como já vimos, cada caracter gráfico de código inferior a 32, ocupa um espaço em dobro na string. Isso se deve a uma particularidade do MSX: como os 31 primeiros caracteres correspondem a códigos de controle do micro ou da impressora, para imprimir um deles na tela, o MSX faz a seguinte transformação:

```
PRINT CHR$(N)= PRINT CHR$(1)+CHR$(64+N)
```

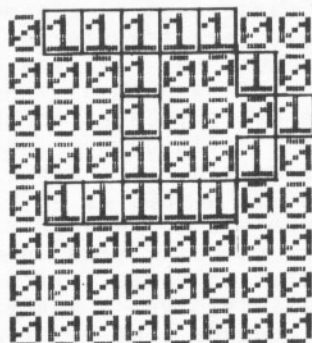
A sub-rotina em questão, simplesmente lê A$ e toda vez que encontra um caracter de código 1, o despreza e subtrai 64 do seguinte.

370-570 Este trecho do programa lê a string A$ (já expurgada dos caracteres 1) e procura a forma do caracter numa tabela para transformá-lo numa sequência de bytes gráficos. Como foi ativada a SCREEN 1, a tabela pesquisada está na VRAM nos endereços de 0 a 2048. Cada 8 bytes desta tabela formam um caracter. Porisso o endereço E definido na linha 400 é o próprio código do caracter multiplicado por 8.

Nas linhas de 370 a 460, A$ é "fatiada" em seus caracteres individuais (C$), o endereço de cada um é procurado na VRAM e os 8 bytes que o formam são passados para a forma binária (linhas 430 e 440). Desta forma fica mais visível o esquema de formação do caracter: o 0 corresponde a um pixel apagado e o 1 a um pixel aceso. Veja a figura 12.3.

Figura 12.3

O próprio programa se encarrega de colocar estes códigos binários (B$) na tela de maneira a visualizar o caracter. Se você não conseguir enxergar o caracter no meio da confusão de 1 e 0, feche um pouco os olhos (tipo "olhar de míope sem óculos") e verá a "sombra" do caracter formada pelos 1's.
O trecho de 480 a 540 se encarrega de fazer outro tipo de "fatiamento": ele pega o primeiro bit (0 ou 1) de cada um dos oito códigos binários do caracter para formar P$, a string a ser enviada para a subrotina de impressão. Note que este fatiamento é feito de baixo para cima na linha 500 (de 7 a 0 STEP -1) pois a impressão ocorre da esquerda para a direita. Isto provoca uma rotação do caracter de 90 graus no sentido horário.

A seguir ele pega o segundo bit dos B$ para formar o próximo P$, e assim sucessivamente até chegar ao "LAésimo". Note que LA pode ser 6 (desprezam-se os dois bits da direita), 7 (despreza-se só o último bit) ou 8 (caracter completo como na SCREEN 1. Veja na figura 12.4 os P$ do caracter A

Figura 12.4

Finalmente, vamos analisar a rotina de impressão que vai da linha 580 à 730: de 590 a 620 os bits 0 e 1 dos P$ enviados para esta rotina por 530, são transformados em códigos 0 (nenhuma impressão) ou WW (4 ou mais agulhas ativadas). Para melhor entender os valores de WW, devemos lembrar que, no modo gráfico, as agulhas são ativadas segundo o padrão binário de WW: onde há 0 a agulha não bate, onde há 1 ela é acionada. Fazendo-se a conversão de decimal par binário dos valores possíveis de WW isto é facilmente percebido:

WW= 170=&B10101010
WW= 204=&B11001100
WW= 240=&B11110000
WW= 255=&B11111111

Em 630 define-se quantas vezes a mesma linha será repetida (largura do pixel), em 650 a impressora é colocada no modo gráfico segundo o padrão de largura já definido em 220 e de 660 a 700 os "macro-pixels são enviados para a impressora.

Na linha 720 é enviado o código 10 (line feed) para avançar o papel. Aliás, toda vez que o programa encontra um P$ cheio de 0's (nenhuma impressão), ele é desviado para esta linha para economizar tempo.

Se você quiser, por exemplo, alterar o programa de maneira a produzir impressão múltipla (várias passadas de cabeça para reforçar o traço),deverá fazer um laço que repita a impressão e só no fim enviar código 10.

Se algo não ficou claro nessa dica, recomendamos que você leia sobre o assunto nos seguintes livros:

APROFUNDANDO-SE NO MSX - cap.6
100 DICAS PARA MSX - Dicas de Impressora

13 EDITOR PARA EMERGÊNCIAS

Esta dica foi criada pensando na nossa tão conhecida "lei de Murphy". Uma das poucas coisas nesse planeta que não nos deixa na mão.

Imagine só aquela noite em que você lembra que tem de entregar, na manhã seguinte, um trabalho escolar ou aquele relatório que o seu chefe pediu para você fazer com "atenção especial".

Você corre para o micro, pega o seu editor de textos predileto e...

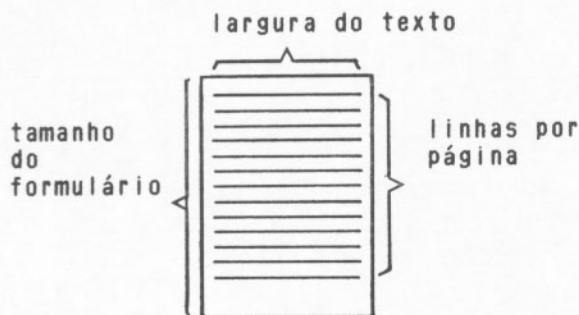
E fica por aí mesmo. Deu "pau" no disco ou na fita. Aí você se lembra que a cópia foi emprestada para um amigo. Corre para o telefone para pedir cópia de volta, mas o telefone toca, toca e ninguém atende.

O que fazer?

Pegar aquela velharia de máquina de escrever que o seu pai usa está fora de cogitação. Afinal, você gastou uma "nota preta" para entrar na era da informática e quando mais você precisa ela te deixa a ver navios.

Elaboramos um "Editor caseiro" ou "Editor de emergência" para ser usado nessas "horas tenebrosas".

O programa (figura 13.2) gera um outro programa BASIC com uma série de LPRINTs. Mas a grande vantagem de usar esse programa para gerar os LPRINTs, são os parâmetros que ele pede durante a execução. Veja na figura 13.1 o formato de um texto impresso.

Figura 13.1

O programa pede o número de caracteres por linha para que os LPRINTs nos quais o texto deverá ser digitado possuam tantos espaços quantos forem necessários para dar a largura do texto a ser digitado.

O número de linhas por página de texto necessário pois as linhas restantes deverão sair em branco, e conterão apenas o comando LPRINT.

O tamanho do formulário serve para sabermos quantos LPRINTs deverão ser repetidos.

O número de páginas irá informar quantas vezes esse processo deverá ser gerado.

Figura 13.2

```
100 INPUT "Quantos caracteres por linha";CL
110 INPUT "Quantas linhas por página";LP
120 INPUT "Tamanho do formulário";TF
130 INPUT "Quantas folhas";FO
140 OPEN "A:edemerg.bas" FOR OUTPUT AS #1
150 L=1000
160 FOR F=1 TO FO
170 FOR T=1 TO TF
180 L$= STR$(L)+"LPRINT"
190 IF T<=LP THEN L$=L$+CHR$(&H22)+STRING
$(CL,32)+CHR$(&H22)
200 PRINT#1,L$
210 L=L+10
220 NEXT T
230 PRINT#1,L"REM FIM DA Página "F
240 L=L+10
250 NEXT F
260 CLOSE
```

Se você possui apenas o gravador cassete, altere a linha 140 para:

```
140 OPEN "CAS:EDEMER" FOR OUTPUT AS #1
```

Após rodar o programa comande:

```
LOAD"EDEMERG.BAS"
```

ou se você possui apenas o gravador cassete:

```
LOAD"CAS:EDMER"
```

Você deverá obter um programa BASIC com várias linhas LPRINT. Preencha os espaços em branco com o texto e rode o programa.

Você deverá obter o texto, página por página na impressora.

Você pode usar esse programa para gerar seqüências de comandos PRINT para ter telas de texto sucessivas.

Para isso basta alterar a linha 180 para:

```
180 L$= STR$(L)+"PRINT"
```

Boa Sorte!!!

14

FILTRO DO ZERO CORTADO

Esta dica nos foi solicitada por um usuário de uma impressora Olívia da Elebra, que ao tirar listagens de seus programas fazia uma tremenda confusão entre a letra "O" e o número "0".

Existem muitas outras impressoras, principalmente importadas, dedicadas ao mercado de usuários que não entendem de informática, apenas usam o computador como um misto de calculadora com máquina de escrever.

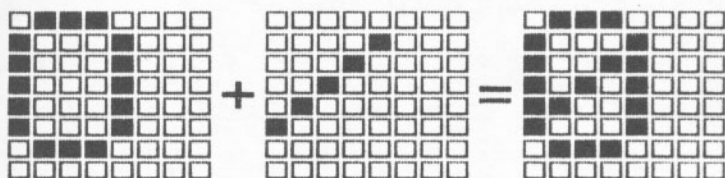
Esse universo de usuários não está acostumado ao tão famoso zero cortado da área computacional. Ficaria até estranho para um advogado ou médico ter impresso em seus relatórios os nossos tão famosos "zerinhos".

Porém, para nós (programadores, hobbystas) eles são imprescindíveis para que não haja confusão na compreensão das listagens. E afinal, estamos tão acostumados com o zero cortado ao ler um programa que uma listagem sem eles fica "meio estranha".

O problema é: como cortar o zero de uma impressora?

Temos em qualquer impressora o caracter "/" de código &H1F, que faz parte da tabela ASCII.

Se combinarmos o caracter do zero (que na impressora não é cortado) com a barra de divisão teremos um zero cortado. Veja na figura 14.1 essa mixagem.

Figura 14.1

Podemos realizar essa mixagem através de um filtro para impressora.

Os filtros para impressora (que são programas em Linguagem de Máquina) operam a partir da alteração do hook HLPTO (&HFFB6) pois toda vez que a rotina LPTOUT

do BIOS é chamada ele também o é.

O registrador A contém o código ASCII do caracter. Portanto basta uma comparação com o código &H20 para sabermos se é ele quem vai ser enviado à impressora.

Caso o código seja diferente, basta um RET (RETurn) para que a execução volte ao interpretador.

Se o caracter a ser enviado for um "0" (código &H30) basta enviar à impressora a seguinte seqüencia:

/ ^H O

a barra serve para cortar o zero, o ^H (lê-se CONTROL-H, caracter 08) é necessário para voltar o carro de impressão e a letra "O" substitui o zero.

Foi escolhida a letra "O" e não o caracter "0" por razões de simplificação do programa filtro.

A hook da rotina LPTOUT é alterada para que o programa funcione, e o próprio programa chama 3 vezes a LPTOUT. Caso um desses caracteres fosse o próprio zero, o filtro operaria de novo, passando a um círculo vicioso.

Veja na figura 14.2 a versão BASIC do programa. A listagem em Assembler encontra-se no Apêndice I.

Figura 14.2

```
1000 DATA 21,0C,D0,22,B7,FF,3E,C3
1010 DATA 32,B6,FF,C9,FE,30,C0,3E
1020 DATA 2F,CD,A5,00,3E,08,CD,A5
1030 DATA 00,3E,4F,CD,A5,00,33,33
1040 DATA C9
1050 FOR L=&HD000 TO &HD020
1060 READ A$
1070 POKE L , VAL("&H"+A$)
1080 NEXT L
1090 DEFUSR=&HC000
1100 POKE 0,USR(0)
1110 LLIST
```

15

FILTRO PARA IMPRESSORAS IMPORTADAS

Os programas de filtro para impressora já foram abordados em várias publicaçes da Aleph, pois é uma grande decepção para quem paga uma verdadeira "fortuna" por uma impressora e não consegue nem imprimir textos corretamente.

Mesmo com todos esses filtros publicados, uma parcela de usuários ficou literalmente a "ver navios", pois eles só funcionam com as impressoras que possuem os caracteres acentuados em suas tabelas de caracteres.

Quem pensou em economizar comprando uma impressora de importabando teve uma bela decepção ao imprimir seus textos.

Na dica anterior, fizemos um filtro para cortar o zero das impressoras que não o fazem, usando os caracteres:

&H1F "/" (barra de divisão)
&H08 "^H" (retorno de carro)
&H4F "O" (letra "O")

Podemos usar a mesma idéia para montarmos os caracteres acentuados. Veja essa mixagem na figura 15.1

Figura 15.1

```
a + ' =á
e + ' =é
i + ' =í
o + ' =ó
u + ' =ú
    |
    └──────> caracter &H27
```

Trocando o caracter &H27 pelo caracter &H7E (til) teremos as letras acentuadas por til, pelo caracter &H60 e &H5E as acentuadas pelos acentos grave e circunflexo.

O "Ç" e o "ç" são facilmente montados com a sobreposição pela vírgula.

```
C + , = Ç
c + , = ç
```

Veja na figura 15.2 o resultado na impressora.

Figura 15.2

ÁÉÍÓÚ
ÃĨÕŨ
ÂÊÔ
ÀÒ
áéíóú
ãĩõũ
âêôû
àò
üçÇ

Em algumas letras o resultado estético final não é lá grande coisa, mas legível. Garanto que é melhor do que passar por analfabeto!

Na figura 15.3 está a listagem BASIC do programa. O fonte, em Assembler, está listado no apêndice I.

Figura 15.3

```
1000 DATA 21,11,D0,22,B7,FF,3E,C3
1010 DATA 32,B6,FF,3E,FF,32,17,F4
1020 DATA C9,32,41,D0,D9,21,B6,FF
1030 DATA 36,C9,3A,41,D0,FE,80,38
1040 DATA 19,D6,80,6F,26,00,29,11
1050 DATA 42,D0,19,7E,CD,A5,00,3E
1060 DATA 08,CD,A5,00,23,7E,CD,A5
1070 DATA 00,E1,21,B6,FF,36,C3,D9
1080 DATA C9,00,43,2C,75,22,65,27
1090 DATA 61,5E,41,27,60,61,22,00
1100 DATA 63,2C,65,5E,49,27,4F,5E
1110 DATA 55,27,41,5E,45,5E,4F,5E
1120 DATA 41,60,45,27,00,00,00,00
1130 DATA 6F,5E,00,00,60,6F,5E,75
1140 DATA 00,00,00,00,00,00,00,55
1150 DATA FF,00,00,00,00,00,00,00
1160 DATA 00,00,61,27,69,27,6F,27
1170 DATA 75,27,00,00,00,00,61,5F
1180 DATA 6F,5F,00,00,00,00,00,00
1190 DATA 00,00,00,00,00,00,00,00
1200 DATA 00,00,41,7E,61,7E,49,7E
1210 DATA 69,7E,4F,7E,6F,7E,55,7E
1220 DATA 75,7E,00
1230 FOR L=&HD000 TO &HD0B2
1240 READ A$:POKE L,VAL("&H"+A$):NEXT
1250 DEFUSR=&HD000: POKE 0,USR(0)
```

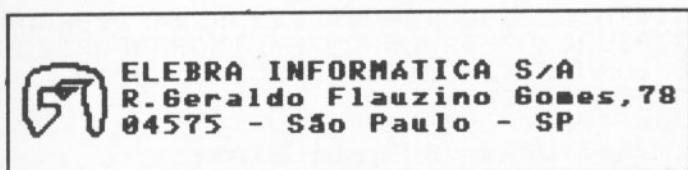
ETIQUETAS COM LOGOTIPO

16

Se você tem uma impressora, certamente teve vontade de tirar etiquetas mais incrementadas e personalizadas. A etiqueta do remetente, por exemplo, pode ter o logotipo de sua firma ou um desenho interessante que o indentifique.

Um exemplo dessas etiquetas está na figura 16.1

Figura 16.1

ELEBRA INFORMÁTICA S/A
R.Geraldo Flauzino Gomes,78
04575 - São Paulo - SP

Para fazer um programa que tire automaticamente essas etiquetas você deve instalar um programa em Linguagem de Máquina que copia a tela do MSX na impressora.

Este programa foi publicado no "CEM DICAS PARA MSX" (pág 127) e no boletim informativo da ALEPH número 7.

Uma vez instalado no micro, ele pode ser gravado na forma binária em disco com o comando:

```
BSAVE"COPYSCR.BIN",&HE000,&HE25F
```

ou em fita com:

```
BSAVE"COPYSC",&HE000,&HE25F
```

Desta forma, toda vez que quisermos chamar o programa para instalá-lo de maneira rápida no micro, basta comandar, a partir do disco:

```
BLOAD"CAS:COPYSCR.BIN",R
```

ou, a partir da fita:

```
BLOAD"COPYSC",R
```

Este programa é ativado toda vez que você pressiona a tecla ESC.

Vejamos como ativá-lo a partir do programa, de maneira a acionar automaticamente no momento certo. Digite o programa da figura 16.2 tomando bastante cuidado com a linha 130.

Figura 16.2

```
100 E$="ELEBRA INFORMÁTICA S/A"
110 F$="R.Geraldo Flauzino Gomes,78"
120 G$="04575 - São Paulo - SP"
130 A$="U15E2U4E4R3E2R4E2R12F2R4F2R3F4D4F
2D15G2D6G2L3H3U7H2U5H2U7E4L15G3D3F3R2E3U2
R3L18G2D9U1E2R7F2D4G10L3H2U2H2U2H2
140 SCREEN 2:OPEN"GRP:" AS #1
150 FOR I=0 TO 146 STEP 73
160 L=20+I:DRAW "BM3,=L;S3XA$;"
170 L=21+I:DRAW "BM3,=L;XA$;"
180 L=20+I:DRAW "BM4,=L;XA$;"
190 FOR J=0 TO 1
200 PRESET(42+J,I):PRINT#1,E$
210 PRESET(42+J,I+10):PRINT#1,F$
220 PRESET(42+J,I+20):PRINT#1,G$
230 NEXT J
240 NEXT I
250 DEFUSR=&HE015:POKE 0,USR(0)
260 LPRINT:LPRINT:LPRINT
270 GOTO 250
```

As linhas de 100 a 120 contém os dizeres da etiqueta. A linha 130 define os comandos a serem utilizados pelo comando DRAW para desenhar a águia estilizada (logotipo).

A linha 140 abre um arquivo na SCREEN 2 para que possamos escrever nela.

O laço 150-240 faz repetir o desenho 3 vezes na tela pulando 73 em 73 pontos. Este foi o valor que encontramos para nossas etiquetas, usando uma impressora MÔNICA PLUS. Se você tiver outra impressora ou outro padrão de etiquetas, deverá pesquisar qual o valor mais conveniente a ser especificado pelo laço em 150.

As linhas de 200 a 220 escrevem os dizeres da

etiqueta e o laço 190-230 faz com que isso seja feito duas vezes para "engrossar" os caracteres.

A linha 250 é o ponto chave do programa: ela ativa a impressora de maneira a copiar o conteúdo da tela nas etiquetas. Obviamente ela não vai funcionar, provocando alguma catástrofe, se o programa de cópia gráfica já não estiver instalado no micro entre os endereços &HE000 e &HE25F.

A linha 260 faz o papel avançar até o começo da etiqueta seguinte (o número de LPRINTs vai depender também do tamanho da etiqueta e da impressora) e a linha 270 torna o processo repetitivo (LOOP infinito), sendo interrompido apenas por CONTROL+STOP.

Se você quiser testar o programa sem impressora, coloque um REM à frente das instruções contidas nas linhas 250 e 260. Mãos à obra: esperamos que a próxima carta que você enviar à ALEPH venha com uma etiqueta personalizada!

USANDO A TABULAÇÃO DA IMPRESSORA

Existe o recurso de tabulação que muitas impressoras possuem, mas que é muito pouco usado. Ele não tem muita utilidade quando se trata de textos a serem impressos, mas quando usamos a impressora de um modo mais profissional, os recursos de tabulação são muito atraentes, visto que aumentam a velocidade de impressão e simplificam a montagem do programa.

O funcionamento é muito parecido com o de um máquina de escrever, só que em vez das colunas para a tabulação serem marcadas com o pressionamento de mais e mais botões, é tudo feito por software.

Para indicar à impressora quais colunas serão "marcadas" pela tabulação, existe o comando (em BASIC):

```
LPRINTCHR$(27)"D"CHR$(T1)CHR$(T2)...CHR$(0)
)
```

onde T1, T2, ... deverão ser os números das colunas nas quais a tabulação deverá ser fixada. Consulte o manual da impressora para se certificar de quantos caracteres podem ser enviados.

Uma vez fixadas as colunas da impressora para a tabulação, basta enviar o caracter 9. A cada envio desse caracter, a impressora avança o carro para próxima posição da tabulação.

Esse recurso é muito bom quando imprimimos listas ou tabelas.

Imagine que queremos uma listagem que apresente nome e telefone de algumas passoas.

Pessoas diferentes possuem nomes de tamanhos diferentes. Então poderemos demarcar, por exemplo, a coluna 25 da impressora para a tabulação (25 caracteres são suficientes para um nome).

Digite e rode o programa da figura 17.1

Figura 17.1

```
10 LPRINT CHR$(27)"D"CHR$(25)CHR$(0)
20 READ NO$,TE$
30 IF NO$="FIM" THEN END
40 LPRINT NO$;CHR$(09);TE$
50 GOTO 20
```

```
60 DATA Solange Almeida,283-4569
70 DATA Fernanda Chaui Petroni,299-5745
80 DATA Sueli Santos,336-5689
90 DATA Rachel Leite Santos,575-8914
100 DATA Paula Andrade,69-5621
110 DATA FIM,XXXXX
```

A parte chave do programa está na linha 10 que "marca" a tabulação e na linha 40 que usa a tabulação imprimindo o caracter 09.

Você obterá um resultado parecido com a figura 17.2

Figura 17.2

```
    Solange Almeida 283-4569
    Fernanda Chaui Petroni  299-5745
    Sueli Santos    336-5689
    Rachel Leite Santos     575-8914
    Paula Andrade   69-5621
```

O resultado obtido não foi o desejado (os telefones não foram impressos na coluna 25). Porque?

Rode o programa da figura 17.3 e veja o resultado no vídeo e na impressora.

Figura 17.3

```
10 FOR L=0 TO 20
20 LPRINTSTRING$(L,65);CHR$(9);"L="L
30 PRINTSTRING$(L,65);CHR$(9);"L="L
40 NEXT L
```

O programa imprime várias seqüencias (de 0 a 20) da letra "A" no vídeo e na impressora. Note que a tabulação no vídeo e na impressora é a mesma, ou seja, de 8 em 8 caracteres.

Isso ocorre porque a rotina do comando LPRINT converte o caracter 09 (TAB) no número de espaços correspondentes. Temos então uma tabulação pré-fixada pelo micro e a tabulação da impressora não pode ser acessada diretamente pelo BASIC.

Para enviarmos o caracter 09, devemos usar uma rotina em linguagem de máquina que chame a rotina LPTOUT ou usar o comando OUT do BASIC (veja a dica

número 18).

Enviar o caracter 09 em linguagem de máquina é mais fácil, pois apenas com um comando teremos a impressora posicionada na coluna que queremos.

Veja a seguir as alterações necessárias para que o programa da figura 17.1 funcione corretamente.

```
1 DATA 3E,09,CD,A5,00,C9
2 FOR L=&HFFF0 TO &HFFF5
3 READ A$
4 POKE L,VAL("&H"+A$):NEXT L
5 DEFUSR=&HFFF0
40 LPRINT NO$;USR(TE$)
```

As linhas de 1 a 4 instalam a rotina em linguagem de máquina a partir do endereço &HFFF0. A linha 5 define o endereço de entrada para a função USR0.

Note que a linha 40 não possue mais referencia ao caracter 09. O comando:

```
USR(TE$)
```

imprime o caracter 9 (através da rotina em L.M.) e também a variável TE$ (telefone). Pois se comandássemos um:

```
USR(0);TE$
```

teriamos um incômodo zero antes do número do telefone.

Essa é também uma boa dica para economizar comandos em seus programas que chamem rotinas em linguagem de máquina.

COMUNICAÇÃO DIRETA COM A IMPRESSORA. 18

Existem vários meios de enviar dados para a impressora. Podemos usar editores de texto, o comando TYPE do MSX-DOS (com o ^P ativado) ou até mesmo com o famoso comando LPRINT do BASIC.

Todos esses comandos chamam a rotina LPTOUT (&H00A5) do BIOS, pois ela é padrão em qualquer MSX. E quanto menos um programa depender do hardware da máquina, maior chance ele terá de rodar em diferentes computadores.

Porém, podemos enviar, via BASIC, qualquer caracter para a impressora com relativa facilidade. E sem a interferência de filtros ou de qualquer rotina da ROM do MSX.

Existem duas portas de I/O do MSX para efetuar a comunicação com a impressora. São elas:

&H90 - Controle
&H91 - Dados

A porta &H91 é usada para enviar o código ASCII do caracter para a impressora. Ao comandar (em BASIC):

```
OUT (&H91),65
```

o caracter ASCII de código 65 (letra A) será enviado para a impressora.

A porta &H91 é um "buffer" de apenas um byte. Qualquer dado que for enviado por essa porta, permanecerá ali até ser alterado ou até o micro ser desligado.

Enviando um dado pela porta &H91, fazemos a informação chegar até a impressora. Mas como devemos fazer para que ela reconheça um novo dado?

É justamente para isso que existe a porta de controle (&H90). Ela facilita a comunicação entre o micro e a impressora. Veja na figura 18.1 a sua divisão.

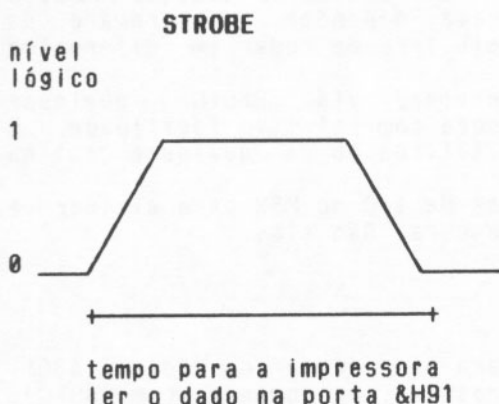
Figura 18.1

O bit de STROBE (bit 0) serve para que o micro comunique a impressora que um novo dado está à sua disposição.

Essa comunicação é feita através de um pulso. É dado um OUT na porta &H90 com o bit de STROBE setado. Algum tempo depois (questão de milisegundos) é dado outro OUT resetando o bit de STROBE. Veja na figura 18.2 o funcionamento do STROBE.

Figura 18.2

nível
lógico

tempo para a impressora
ler o dado na porta &H91

É durante o tempo do STROBE em nível lógico 1 que a impressora realiza a leitura do dado que está na porta &H91 (porta de dados).

O programa da figura 18.3 é um exemplo de envio para a impressora através do comando OUT.

Figura 18.3

```
10 READ D$
20 OUT(&H91),ASC(D$)
30 FOR L=0 TO 80
40 OUT(&H90),1:OUT(&H90),0
50 NEXT L
60 LPRINT
70 READ D$
80 IF D$="FIM" THEN END ELSE GOTO 20
90 DATA A,L,E,P,H,FIM
```

Note que o código do caracter é enviado na linha 20, e no loop das linhas 30-50 o STROBE é variado 80 vezes para que a impresora reconheça o envio de 80 letras iguais, pois o código fica permanentemente no buffer da porta &H91 enquanto não for alterado.

Apenas na linha 60 é enviado um LPRINT para que a impressora avançe uma linha.

A impressora é um periférico muito lento quando comparado com o drive ou até mesmo com o cassete; e o micro é capaz de mandar informações em uma velocidade muito maior do que ela pode imprimir.

Os modelos mais sofisticados de impressora possuem uma memória RAM interna (buffer de impressora) que pode chegar a até vários Kbytes. Mas se a informação enviada pelo micro for muita? até mesmo para o buffer da impresora?

Quando isso acontece, a impressora envia para a porta de controle o bit de BUSY (ocupado) setado a fim de que o micro pare de enviar dados (pois ela não poderá reconhecê-los).

As operações de BUSY ocorrem principalmente nos seguintes casos:

Buffer cheio
Avanço de formulário
Retorno de carro
impressora "fora de linha"
beep
defeito interno
Aquecimento extremo.

mas podem variar de acordo com o modelo de impressora.

Quando o cabo não está conectado ao micro ou a impressora, o bit de BUSY também estará setado. Esta é uma boa maneira de saber se existe uma impressora conectada ao micro.

Digite o programa da figura 18.4:

Figura 18.4

```
20 CLS:L=32
30 A=INP(&H90) AND &B00000010
40 IF A=2 THEN BEEP:LOCATE 5,10:PRINT"IMP
RESSORA !!!":RUN
50 OUT (&H91),66
60 OUT (&H90),1
```

```
70 OUT (&H90),0
75 L=L+1
80 A=INP(&H90)
85 IF L=126 THEN LPRINT: L=32
90 PRINT RIGHT$("0000000"+BIN$(A),8);L
100 IF (A AND 2)=2 THEN PRINT"AGUARDANDO
IMPRESSORA!":GOTO 80
110 GOTO 60
```

Rode-o programa com o cabo da impressora desconectado.

Você deverá observar uma mensagem gerada pelas linhas 30 e 40 indicando que a impressora não está conectada.

Rode-o novamente, mas com o cabo da impressora conectado.

As linhas de 50 a 70 enviam um caracter (dado pela variável L).

O bit de BUSY é lido na linha 80, impresso no vídeo na linha 90 e testado na linha 100. Caso esteja setado, a mensagem de espera é impressa no vídeo.

Infelizmente não há como detectar se a impressora está ligada ou não. Pois o bit de BUSY, nesse caso, fica ressetado. Rode o programa com a impressora desligada e conectada ao micro.

19

Além do comando PLAY A$ que, quando executado, toca os códigos contidos em A$, o BASIC MSX tem função PLAY(n) que permite detectar se um determinado canal de som está ativado.

O valor de n está associado aos canais de som conforme a tabela a seguir.

n	canal
0	qualquer
1	A
2	B
3	C

Quando o canal está ativo, a função assume o valor -1, caso contrário ela devolve o valor 0.

Rode o programa da figura 19.1 para perceber seu efeito.

Figura - 19.1

```
100 PLAY"T32V1504","T32V1504","T32V1504"
110 SCREEN 0
120 LOCATE 0,5:PRINT" CANAL:           A     B
    C  (A ou B ou C)"
130 PLAY"C2.","E2","G4"
140 FOR I=0 TO 220
150 LOCATE 10,7:PRINTTAB(12);PLAY(1);TAB
(17);PLAY(2);TAB(22);PLAY(3);TAB(31);PLAY(0)
160 NEXT I
170 GOTO 130
```

Para perceber sua utilidade, vamos elaborar um curto programa: digamos que, por motivos didáticos, você queira tocar uma música no micro enquanto o códigos de cada compasso aperecem na tela.

Digite o programa da figura 19.2 e rode-o.

Figura - 19.2

```
100 SCREEN 0
110 PLAY"s0m9000t200o4"
120 DATA e2.,d4c4d4,c2.,c2r4,o5c2.,o4a4f
```

```
4a4,g2.,g2r4,o5c4o4a4o5c4,o4b4g4b4,a4f4a
4,g2.,e2.,d4c4d4,c2.
130 FOR I=1 TO 15
140 READ  P$
150 PRINT P$
160 PLAY  P$
180 NEXT I
```

A linha 110 define o envelope e o andamento do canal A e a 120 contêm os códigos a serem tocados.

O laço 130-180 lê os códigos da linha DATA, imprime cada compasso na tela e os toca.

Como o MSX tem um processador de som com um pouquinho de inteligência (o PSG) o processamento do programa em BASIC (que determina o que é impresso na tela) e a emissão do som ocorrem de maneira quase independente, provocando uma falta total de sincronia.

Você deve ter notado que o cursor é liberado com Ok quando a melodia ainda está tocando! O efeito "didático" de mostrar cada compasso tocado foi para o espaço!

Experimente, agora, inserir a linha:

```
170 IF PLAY(0) THEN 170
```

Enquanto algum canal estiver ativo, o PLAY(0) assume o valor -1 (que para o micro significa "condição verdadeira") e o programa em BASIC fica executando a linha 170 até o final da execução total do compasso em questão.

Rode o programa e veja como agora tudo sincronizou!

Uma sugestão de utilização desta dica é você associá-la com a animação de sprites (Veja no CEM DICAS PARA MSX!) e criar um personagem que fique dançando ao som de uma música de maneira sincronizada!

RESPOSTA SONORA NO TECLADO

20

O sistema de entrada e saída do MSX é extrememente versátil pois as rotinas mais importantes podem ser facilmente modificadas com apenas alguns "pokes" nos lugares certos. Ao se chamar uma rotina de E/S qualquer (leitura de teclado, impressão na tela, etc), a ROM checa certos endereços de uma área chamada HOOKS (em português, ganchos). Cada rotina possui 5 bytes de hook. A rotina CHGET (que lê uma tecla) possui o hook entre os endereços &HFDC2 e &HFDC6. Este hook, como todos os outros, está preenchido com 5 bytes &HC9 (RET). Se alterarmos estes bytes, podemos mudar as características da leitura de teclas.

Experimente digitar:

```
POKE&HFDC2,195:POKE&HFDC3,192:POKE &HFDC4,0
```

Aperte então algumas teclas. Deverá soar um "beep" a cada tecla apertada. Para voltar ao normal, comande:

```
POKE &HFDC2,201
```

21 USANDO O MOUSE COM PROGRAMAS EM BASIC

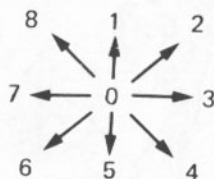
Quando compramos um mouse, geralmente ele vem acompanhado de programas de aplicação. Entretanto, às vezes é desejável poder usá-lo em outros programas criados pelo usuário.

Para suprir isso, o mouse para MSX tem a capacidade de simular o joystick e o touch-pad padrão MSX. Assim, os comandos STICK, STRIG e PAD podem ser usados com o mouse.

Para simular um joystick, basta manter pressionado o botão esquerdo do mouse ao ligar a força do micro. Note que o RESET não é suficiente: necessário desligar e ligar o micro.

A simulação do touch-pad é feita pressionando-se o botão direito.

A diferença entre o uso do mouse como joystick e touch-pad, é que no primeiro caso temos disponível a direção de seu deslocamento pela função STICK(n). O aperto dos botões é detectado pela função STRIG(n). Veja na figura 21.1 o retorno da função STICK para cada direção.

Figura 21.1

Já o mouse como touch-pad tem disponíveis as coordenadas X e Y de sua posição e pode ser lido pela função PAD(n). O programa da figura 21.2 mostra um exemplo do funcionamento do mouse como touch-pad.

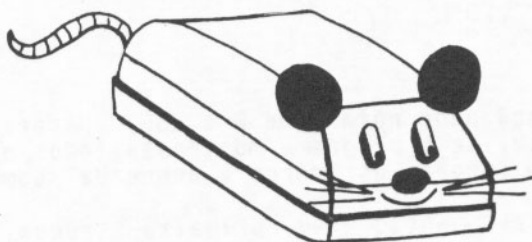
Digite o programa e rode-o com o mouse ligado. Pressione os botões do mouse e movimente-o.

Observe os valores de cada n da função PAD.

Figura 21.2

```
10 CLS
20 PRINT"n"CHR$(9)"PAD(n)"
30 LOCATE 0,2
40 FOR L= 0 TO 3
50 PRINTL CHR$(9) PAD(L)"   "
60 NEXT
70 GOTO 30
```

Experimente o mouse com jogos para joystick Difícil, não? Por outro lado, o mouse (no modo touch-pad) serve com programas com EDDY-2 e Graphic Master.

22 O OPERADOR LóGICO NOT

Conforme você for evoluindo como programador, sentirá cada vez mais a necessidade de abandonar a notação decimal. O uso da notação binária e hexadecimal o aproximam mais da lógica do micro, tornando rápidas e simples operações que, de outra forma, seriam complexas.

Cada byte do micro é constituido por oito bits que podem assumir o valor 0 ou 1.

O conteúdo de um byte, em binário, vai de &B00000000 (0) a &B11111111 (255), podendo assumir 256 configurações diferentes.

Para ver todas essas configurações, rode o programa da figura 22.1.

Figura 22.1

```
100 SCREEN 1
120 FOR I=0 TO 255
130 B$=BIN$(I)
140 PRINT I,B$
150 NEXT I
```

Como você pode notar, de 0 a 255. Ficaria muito melhor, porém, se o programa mostrasse todos os 8 bits do byte, sem ignorar os "zeros à esquerda" como faz a função BIN$.

Aqui vai, então, um primeiro truque: basta emendar à esquerda do valor dado pelo BIN$ 7 zeros e depois pegar os 8 algarismos da direita (7 zeros porque a função BIN$ retorna, no mínimo um algarismo.). Altere a linha 130 para:

```
130 B$=RIGHT$("0000000"+BIN$(I),8)
```

e você verá, listados na tela, todos os 8 bits, inclusive os "zeros à esquerda".

Vamos agora aprender a manipular esses oito bits usando um pouco de algebra booleana (não se assuste que a coisa é simples!).

Digamos, por exemplo, que você queira redefinir alguns caracteres na SCREEN 0 de maneira a que apareçam em "negativo". A tabela de formação dos

caracteres na SCREEN 0 ocupa dois Kbytes da VRAM a partir do endereço 2048.

Sabemos que, usando o operador lógico NOT, todos os bits 0 são transformados em 1 e vice-versa. Digite essa nova versão do programa anterior para ver o efeito do NOT.

Figura 22.2

```
100 SCREEN 1:INTERVAL ON
110 LOCATE 0,8
120 PRINT" I        BIN$(I)   BIN$(NOT(I))"
130 FOR I=0 TO 255
140 ON INTERVAL=20 GOSUB 170
150 GOTO 150
160 NEXT I
170 B$=RIGHT$("0000000"+BIN$(I),8)
180 C$=RIGHT$("0000000"+BIN$(NOT(I)),8)
190 LOCATE 0,10
200 PRINT USING"###";I;
210 PRINT TAB(7);B$;TAB(16);C$
220 RETURN 160
```

Agora que você já entendeu como funciona o NOT, vamos pegar os bytes que definem a forma dos caracteres dos algarismos de 0 a 9 e vamos "invertê-los" com este operador lógico.

Digite o programa da figura 22.3 e veja o seu efeito.

Figura 22.3

```
100 SCREEN 0
110 FOR I= 0 TO 255
120 E=2048+8*I
130 FOR K=0 TO 7
140 CH=VPEEK(E+K)
150 CN=NOT(CH)
160 B$=BIN$(CN)
170 A$=RIGHT$("0000000"+B$,8)
180 A=VAL("&B"+A$)
190 VPOKE (E+K),A
200 NEXT K
210 NEXT I
220 LIST
```

Obviamente, se você quiser esse mesmo tipo de efeito na SCREEN 1, basta alterar as linhas 100 e 120 para:

```
100 SCREEN 1
120 E=0+8*I
```

pois na SCREEN 1 a tabela de formação dos caracteres começa na posição 0 da VRAM.

Se ao invés de alterar os algarismos, você quisesse, por exemplo, "inverter" todas as letra maiúsculas, bastaria alterar a linha 110 para:

```
110 FOR I=ASC("A") TO ASC("Z")
```

Faça esta alteração e veja seu efeito.

Se quiser inverter todos os caracteres, altere a linha 110 para:

```
110 FOR I=0 TO 255
```

Obviamente, como o CHR$(32) (espaço) também é invertido, isso equivaleria a inverter o COLOR da tela. Se quiser insira a linha;

```
115 IF I=32 THEN 210
```

para não inverter o espaço.

Finalmente, encerrando essa dica, se você quiser acompanhar o efeito do NOT ao longo da tabela de caracteres, acrescente estas linhas a seu programa:

```
105 GOSUB 300
300 FOR X=0 TO 255
310 IF X<32 THEN PRINT CHR$(1)+CHR$(X+64)
;ELSE PRINT CHR$(X);
320 NEXT X
330 RETURN
```

O OPERADOR LóGICO AND

23

O operador lógico AND trabalha da seguinte forma: comparando dois bits ele só resulta em 1 se ambos forem iguais a 1. Se forem diferentes o resultado é 0.

Digite o programa da figura 23.1 e rode-o várias vezes até se familiarizar com os efeitos do AND.

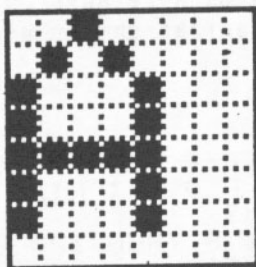
Figura 23.1

```
100 SCREEN 1,1
110 PRINT "DIGITE 8 ALGARISMOS 0 OU 1    P
ARA DEFINIR O X E O Y":PRINT
120 INPUT "X=";X$
130 IF LEN (X$)<>8 THEN 120
140 X=VAL("&B"+X$)
150 INPUT "Y=";Y$
160 IF LEN (Y$)<>8 THEN 150
170 Y=VAL("&B"+Y$)
180 SPRITE$(1)=CHR$(255)+CHR$(X)+CHR$(255
)+STRING$(5,0)
190 SPRITE$(2)=CHR$(255)+CHR$(Y)+CHR$(255
)+STRING$(5,0)
200 Z=X AND Y
210 SPRITE$(3)=CHR$(255)+CHR$(Z)+CHR$(255
)+STRING$(5,0)
220 CLS
230 LOCATE 0,2
240 PRINT "X=";X;TAB(8);"=";X$
250 PUT SPRITE 1,(158,16),15,1
260 LOCATE 0,4
270 PRINT "Y=";Y;TAB(8);"=";Y$
280 PUT SPRITE 2,(158,32),15,2
290 LOCATE 0,6
300 PRINT "X AND Y ="
310 Z$=RIGHT$("0000000"+BIN$(Z),8)
320 PUT SPRITE 3,(158,48),15,3
330 LOCATE 9,6
340 PRINT Z$
350 PRINT:PRINT"APERTE QUALQUER TECLA"
360 A$=INPUT$(1)
370 RUN
```

Imagine que você queira redefinir uma tabela de letras mais "estreitas" (veja a dica 46-DIR EM SCREEN

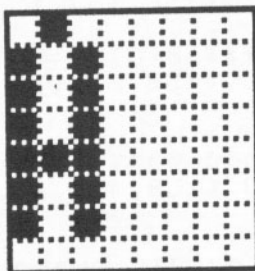
2 para ver um uso desta tabela).

Como você sabe, os caracteres do MSX são definidos em uma tabela 8X8. As letras, porém, são definidas numa tabela 5X8 (figura 23.2) pois, na SCREEN 0 são plotadas apenas as 6 colunas da esquerda para podermos colocar 40 letras (com uma coluna vazia entre eles para que as letras não emendem) por linha de tela.

Figura 23.2

Usando a SCREEN 2, podemos escrever até 64 letras por linha de tela se usarmos caracteres que usem apenas 3 colunas (e uma vazia para separar uma letra da outra).

Para obter esta tabela de letras estreitas, podemos redefinir os caracteres um a um, usando o programa da dica 2-B do livro "CEM DICAS PARA MSX". Isso, porém, daria muito trabalho. Será que não podemos achar um algorítmo que faça isso automaticamente?

Veja a figura 23.3, onde está um caracter "estreito" para descobrir uma regra que o forme a partir do original "largo".

Figura 23.3

Como você deve ter notado, a primeira coluna será preenchida se, no original, houver bit 1 na primeira ou segunda coluna.

A segunda coluna será preenchida se houver bit 1 na terceira coluna do original.

A terceira coluna do caracter "estreito" será preenchida se houver bit "aceso" (1) na quarta e quinta coluna do original.

Usando o operador lógico AND, é fácil transformar esta regra num programa que estreite caractere automáticamente: digite (e analise!) o programa da figura 23.4 e veja seu efeito.

Figura 23.4

```
100 SCREEN 0
110 FOR F=2048+32*8 TO 2048+255*8
120  A=VPEEK(F)
130  B=A AND &B00011000
140  C=A AND &B11000000
150  D=A AND &B00100000
160  IF B<>0 THEN B=&B00100000
170  IF C<>0 THEN C=&B10000000
180  IF D<>0 THEN D=&B01000000
190  VPOKE F, (B OR C OR D)
200 NEXT F
```

Obviamente, se você quiser trabalhar na SCREEN 1, deverá alterar as linhas 100 e 110 para:

```
100 SCREEN 1
110 FOR F=32*8 TO 255*8
```

Veja na dica 47-TABELAS DE CARACTERES como passar essa tabela da RAM de maneira a que ela possa ser usada na SCREEN 2.

24 O OPERADOR LóGICO OR

Para entender melhor esta dica, é conveniente ler antes a dica 22 sobre o operador lógico NOT.

Digite o programa da figura 24.1 para entender a função do operador OR: quando fazemos a operação OR entre dois bits, o resultado será 1 se pelo menos um deles for igual a 1. Só será 0 se ambos forem 0.

Figura 24.1

```
100 SCREEN 1,1
110 PRINT "DIGITE 8 ALGARISMOS 0 OU 1  P
ARA DEFINIR O X E O Y":PRINT
120 INPUT "X=";X$
130 IF LEN (X$)<>8 THEN 120
140 X=VAL("&B"+X$)
150 INPUT "Y=";Y$
160 IF LEN (Y$)<>8 THEN 150
170 Y=VAL("&B"+Y$)
180 SPRITE$(1)=CHR$(255)+CHR$(X)+CHR$(255
)+STRING$(5,0)
190 SPRITE$(2)=CHR$(255)+CHR$(Y)+CHR$(255
)+STRING$(5,0)
200 Z=X OR Y
210 CLS
220 LOCATE 0,2
230 PRINT "X=";X;TAB(8);"=";X$
240 PUT SPRITE 1,(158,16),15,1
250 LOCATE 0,4
260 PRINT "Y=";Y;TAB(8);"=";Y$
270 PUT SPRITE 2,(158,32),15,2
280 LOCATE 0,6
290 PRINT "X OR Y  ="
300 Z$=RIGHT$("0000000"+BIN$(Z),8)
310 FOR T=0 TO 1000:NEXT T
320 FOR I=16 TO 32 STEP .1
330 PUT SPRITE 5,(158,I),15,1
340 NEXT I
350 FOR I=32 TO 48 STEP .1
360 PUT SPRITE 5,(158,I),15,1
370 PUT SPRITE 6,(158,I),15,2
380 NEXT I
390 LOCATE 9,6
400 PRINT Z$
```

```
410 PRINT:PRINT"APERTE QUALQUER TECLA"
420 A$=INPUT$(1)
430 RUN
```

Rode o programa várias vezes, usando diversas combinações de 0 e 1, até você se familiarizar completamente com o funcionamento do operador OR.

Agora, antes de fazermos uma aplicação prática do que aprendemos, altere as linhas 150 e 160 do programa anterior para:

```
150 Y=X\2
160 Y$=RIGHT$("0000000"+BIN$(Y),8)
```

e eliminando a linha 170.

Rode o programa e, quando for solicitado o valor de X entre com

```
00001000
```

Como você pode notar, a divisão inteira de X por 2 (Y=X\2) implica num deslocamento dos bits de uma casa para direita (se a divisão fosse por 4, o deslocamento seria de 2 casas, por 8 de 3, e assim por diante).

Vamos aproveitar isso para redefinir a tabela de caracteres de maneira a torná-los mais "grossos". Digite o programa da figura 24.2 e você verá uma interessante aplicação do OR.

Figura 24.2

```
100 SCREEN1
110 FOR I=0 TO 255
120 IF I<32 THEN PRINT CHR$(1)+CHR$(I+64)
; ELSE PRINT CHR$(I);
130 NEXT I
140 FOR I=0 TO 2047
150 A=VPEEK(I)
160 B=A\2
170 C=A OR B
180 VPOKE I,C
190 NEXT I
200 LIST
```

25 O OPERADOR LóGICO XOR

O operador XOR, ao comparar dois bits, devolve o valor 0 se eles forem iguais ou 1 se forem diferentes.

Para entender seu funcionamento e uma aplicação prática, digite o programa da figura 25.1.

Figura 25.1

```
100 SCREEN 1
110 FOR I=0 TO 255
120 IF I<32 THEN PRINT CHR$(1)+CHR$(I+64)
; ELSE PRINT CHR$(I);
130 NEXT I
140 FOR I=0 TO 2046 STEP8
150 FOR G=7 TO 1 STEP-1
160 E=I+G
170 A=VPEEK(E)
180 B=VPEEK(E-1)
190 C= A OR B
200 VPOKE E,C
210 NEXT G
220 NEXT I
230 FOR I=0 TO 2047
240 A=VPEEK(I)
250 B=A\2
270 C=A OR B
280 VPOKE I,C
290 NEXT I
300 FOR I=0 TO 2047
310 A=PEEK(7103+I)
320 B=VPEEK(I)
330 C= A XOR B
340 VPOKE I,C
350 NEXT I
```

As linhas de 100 a 130 simplesmente listam a tabela de caracteres residente (da ROM) na tela.

De 140 a 220 estamos pegando os caracteres e fazendo a superposição de cada um deles com ele próprio (OR) deslocado de um pixel para baixo. Conforme o programa for rodando, você verá os caracteres ficarem mais grossos na vertical.

Quando essa fase termina, o programa executa as

linhas de 230 a 290 que engrossam o caracter na horizontal (você já viu isso na dica anterior que trata do operador OR).

Feito isso, entramos na parte interessante: a linha 310 vai pegar a formação do caracter na ROM (e não na VRAM) e executa um XOR com o caracter "gordo" da VRAM (veja as linhas 320 a 330).

O que vai acontecer? O que houver em comum entre o caracter original da ROM e o "gordo" da VRAM é apagado, gerando letras tipo sombra (figura 25.2).

Figura 25.2

Se você ainda não está
recebendo gratuitamente
o Boletim Informativo
da ALEPH, mande seu no-
me e endereço (com CEP
correto!) para:

O EDITORA ALEPH
O C.P. 20 707
O CEP 01498
O S.PAULO - SP
O (011) 843-3202 ▶

Se você quiser outro tipo de tabela, pode digitar o programa da figura 25.3. (Note que ele é quase igual ao anterior bastando alterar as linhas 270 e 310 e acrescentar a 255.

Figura 25.3

```
100 SCREEN 1
110 FOR I=0 TO 255
120 IF I<32 THEN PRINT CHR$(1)+CHR$(I+64)
; ELSE PRINT CHR$(I);
130 NEXT I
140 FOR I=0 TO 2046 STEP8
150 FOR G=7 TO 1 STEP-1
160 E=I+G
170 A=VPEEK(E)
```

```
180 B=VPEEK(E-1)
190 C= A OR B
200 VPOKE E,C
210 NEXT G
220 NEXT I
230 FOR I=0 TO 2047
240 A=VPEEK(I)
250 B=A\2
255 D=B\2
270 C=A OR B OR D
280 VPOKE I,C
290 NEXT I
300 FOR I=0 TO 2047
310 A=PEEK(7103+I)\2
320 B=VPEEK(I)
330 C= A XOR B
340 VPOKE I,C
350 NEXT I
```

Neste caso estamos engrossando o caracter em dobro na vertical e em triplo na horizontal. Antes de darmos o XOR com o original que está na ROM, ele é deslocado de um pixel para a direita (divisão inteira por 2) de maneira a "esvaziar" a parte central da lera "gorda".

Veja o resultado na figura 25.4

Figura 25.4

```
ESTE TEXTO FOI ESCRITO
COM OS CARACTERES REDE-
FINIDOS PELO XOR:
           *
Não fique isolado! Mande
seu NOME e ENDEREÇO (com
CEP correto!) para:
EDITORA ALEPH - CP 20707
CEP 01498 - S.Paulo - SP
e receba gratuitamente
seu BOLETIM INFORMATIVO.
* Passe esta mensagem *
```

Veremos, na dica 47, como gravar qualquer tabela redefinida e como utilizá-la na SCREEN 2.

Se você quiser que os programas dessa dica rodem na SCREEN 0, altere, alem do comando SCREEN o endereço da VRAM onde o programa lê e redefine os bytes de formação dos caracteres. Lembre-se que essa tabela possui 2048 bytes, começa no endereço 2048 para a SCREEN 0 e no endereço 0 para a SCREEN 1.

26 ITALIZANDO OS CARACTERES

Ao contrário do que alguns leitores poderiam pensar, "italizar" um set de caracteres não significa obrigar o micro a escrever coisas do tipo:"mamma mia", "spaghetti" ou "porca miséria"!

Em linguagem tipográfica, o "itálico" representa uma variação de uma família de caracteres, na qual eles aparecem ligeiramente deitados.

Podemos ter o "itálico extrovertido" (uma espécie de redundância) no qual os caracteres estão deitados para direita, simulando a caligrafia de pessoas extrovertidas. Podemos ter também o "itálico tímido", com caracteres inclinados para a esquerda, como na caligrafia de pessoas introvertidas.

Vamos, inicialmente , fazer o micro colocar toda sua tabela de carcteres na SCREEN 0. Digite o programa da figura 26.1 e, enquanto ele roda, vamos à sua análise.

Figura.26.1

```
100 SCREEN 0:WIDTH 40
110 FOR I=0 TO 255 STEP 16
120  FOR K=0 TO 15
130   C=I+K
140   IF C<32 THEN PRINT CHR$(1)+CHR$(C+6
4);" ";
150   IF C>31 AND C<>127 THEN PRINT CHR$(
C);" ";
160   IF C=127 THEN PRINT"  ";
170  VPOKE 0+40*7+30,127
180  NEXT K
190  PRINT
200 NEXT I
```

O laço 110-200 coloca os 255 caracteres na tela, em filas de 16, separados por um espaço vazio. A linha 190 provoca o salto de linha e a 140 permite a impressão na tela dos caracteres de código menor que 32, normalmente usados como caracteres de controle.

Se você quiser, por exemplo, imprimir o caracter 7 na tela (um "ponto" deslocado para cima) e comandar:

```
PRINT CHR$(7)
```

ao invés de obter o caracter desejado, conseguirá apenas um "beep". Isto porque o CHR$ (7) é reservado para o código ASCII "BEL" ou seja, toque a campainha! Você obterá o mesmo efeito se digitar, simultaneamente CONTROL + G.

Para se convencer disso comande (no modo direto)

```
A$=INPUT$ (1)
```
(e RETURN)

A seguir tecle control+G e comande:

```
PRINT A$
```

você ouvirá a campainha!

Se comandar:

```
PRINT ASC (A$)
```

você obterá "7", que é o código do "beep" ouvido.

Sim, mas como obter o "ponto" desejado e não o sininho? Basta lembrar que, para imprimir na tela caracteres de código N com N menor que 32, devemos comandar:

```
PRINT CHR$(1) CHR$ (N+64)
```

que é o que faz a linha 140.

Existe, porém em outro caracter de controle além desse 32: é o de código 127, correspondente ao "trianguloz inho" (△). Ele equivale ao Back Space (BS).

Experimente comandar, no modo direto:

```
PRINT "ABCD" (e RETURN)
```

você deve obter:

```
ABCD
```

comande agora:

```
PRINT "ABCD" + CHR$(127)
```
(e RETURN)

você obterá:

```
ABC
```

pois o CHR$ (127) equivale a apertar a tecla BS do Expert ou << do HOTBIT.

Assim sendo, na linha 160, substituimos o CHR$ (127) por um espaço vazio (para não desalinhar a tabela) e na linha 170 o colocamos na tela com um VPOKE).

Note que, na SCREEN 0, a tabela dos caracteres impressos na tela começa em 0 (1ª posição) e o nosso caracter deve estar na 30ª posição da 8ª linha de 40 caracteres:

$$0+40*7+30$$

Usando todas essas dicas, você deve obter a tabela da figura 26.2 (note que alguns caracteres gráficos saem "cortados" pois na SCREEN 0 são mostrados apenas os 6 primeiros pontos horizontais de cada caracter).

Figura.26.2

Vamos agora "italizar" de maneira extrovertida a nossa tabela. Acrescente ao seu programa original (da figura 26.1) as linhas listadas na figura 26.3 .

Figura.26.3

```
300 FOR F=0 TO 255
310  H=2048+8*F
320  FOR G=0 TO 7
330    I=H+G
340    A=VPEEK(I)
```

```
350     IF G<4 THEN VPOKE I,A/2
360   NEXT G
370 NEXT F
```

Observe o programa rodando, para perceber melhor seu efeito e entender a explicação que vem a seguir.

Na linha 300 vamos buscar todos os caracteres, cujo código passa a ser F.

Como a tabela que dá a formação dos caracteres na SCREEN 0 começa em 2048 da VRAM e, para cada caracter, temos 8 bytes que definem sua forma, o caracter F terá o início de sua matriz começando em:

```
2048+8*F
```

A linha 320 pega os 8 bytes de cada caracter e os submete ao seguinte algoritmo. Se for um byte da parte superior do caracter (3 primeiros linhas de formação G>4) o valor do byte é deslocado de um bit para direita (A/2).

Lembre-se que , quando dividimos um byte por 2, todos seus bits deslocam-se de uma casa para direita:

```
11001011\2 = 1100101
```

Isto é análogo, na notação decimal, a dividir um número por 10; todos seus dígitos se deslocam de uma casa para direita. Experimente comandar, no modo direto:

```
PRINT 12768\10
```

(o sinal de divisão inteira é obtido,no Expert, por LGRA+ "/" ou, no HOTBIT,diretamante na tecla "\").
Você deverá obter:

```
1276
```

ou seja, o último algarismo da direita, foi eliminado e todos os outros se deslocaram de uma casa.

Sua tabela de caracteres deverá assumir, na tela, o aspecto da figura 26.4

Figura.26.4

E, para italizar ao contrário? Antes de consultar a listagem da figura 26.5 pense um pouco alterações que você faria no programa original.

Figura.26.5

```
100 SCREEN 0:WIDTH 40
110 FOR I=0 TO 255 STEP 16
120  FOR K=0 TO 15
130   C=I+K
140   IF C<32 THEN PRINT CHR$(1)+CHR$(C+6
4);" ";
150   IF C>31 AND C<>127 THEN PRINT CHR$(
C);" ";
160   IF C=127 THEN PRINT"  ";
170  VPOKE 0+40*7+30,127
180  NEXT K
190  PRINT
200 NEXT I
300 FOR F=0 TO 255
310  H=2048+8*F
320  FOR G=0 TO 7
330    I=H+G
340    A=VPEEK(I)\2:VPOKE I,A
350    IF G<4 THEN VPOKE I,A*2
360  NEXT G
370 NEXT F
```

Isso mesmo! Na linha 340 deslocamos todos os bytes de um bit para direita (\2) e na linha 350 redeslocamos os 3 primeiros bytes para esquerda (A*2). A tabela que você obtem na tela, agora tem o aspecto da fig. 26.6

Figura. 26.6

Se quisermos adaptar este algoritmo para a SCREEN 1, devemos lembrar de algumas particularidades, antes de digitar o programa da fig. 26.7:

A) O máximo WIDTH na SCREEN 1 é 32 (linha 100)

B) A tabela de localização de caracteres começa em 6144 da VRAM e cada linha tem 32 caracteres (linha 170).

C) Não há necessidade de se pular uma linha ao fim de 32 caracteres (linha 190 eliminada) .

D) A tabela de formação de caracteres começa, na SCREEN 1, no endereço 0 da VRAM (linha 310) .

E) Como temos 8 pontos, podemos "italizar" mais a parte superior do caracter (linhas 350 a 380) . Lembre-se que, se dividindo por 2, deslocamos de uma casa para direita, dividindo por 4 deslocaremos de 2, por 8 de 3 e por 16 de 4!

Figura. 26.7

```
100 SCREEN 1:WIDTH 32
110 FOR I=0 TO 255 STEP 16
120  FOR K=0 TO 15
```

```
130    C=I+K
140    IF C<32 THEN PRINT CHR$(1)+CHR$(C+6
4);" ";
150    IF C>31 AND C<>127 THEN PRINT CHR$(C);" ";
160    IF C=127 THEN PRINT" ";
170   VPOKE 6144+32*7+30,127
180   NEXT K
200 NEXT I
300 FOR F=0 TO 255
310  H=8*F
320  FOR G=0 TO 7
330    I=H+G
340    A=VPEEK(I)
350    IF G<1 THEN VPOKE I,A/16:GOTO390
360    IF G<2 THEN VPOKE I,A/8:GOTO390
370    IF G<3 THEN VPOKE I,A/4:GOTO390
380    IF G<4 THEN VPOKE I,A/2
390  NEXT G
400 NEXT F
```

Rode este programa e você deverá obter a tabela de figura 26.8

Figura 26.8

Para exemplificar, o texto da figura 26.9 foi obtido usando o programa de cópia gráfica do "100

DICAS PARA MSX" em cima de uma tela gerada na SCREEN 1 com o programa da figura 26.7.

Figura. 26.9

Editora Aleph

C.P. 20 707

CEP 01495

São Paulo SP

(011) 543-3202

Finalizando, se voçe quiser saber como guardar todas estes tabelas na RAM do seu micro e adotá-las no meio de um programa, leia a dica 47.

27 TABELAS DE CARACTERES EM ASSEMBLER

Nas dicas de 22 a 26 você viu como gerar tabelas de caracteres em BASIC, através de algoritmos relativamente simples.

Nesta dica pegamos alguns destes algoritmos e os passamos para Linguagem de Máquina de maneira a tornar o processo muito mais rápido.

Para os que estão familiarizados apenas com, a linguagem BASIC, elaboramos um programa "corregador" listado na figura 27.1. Para os "escovadores de bytes", listamos o programa fonte, em Assembly, no Apêndice 1. Para análise de seu funcionamento aconselhamos, também, a leitura das dicas citadas.

Figura.27.1

```
10 DATA FE,02,C0,23,23,7E,FE,01
11 DATA 20,3B,CD,FB,C0,01,00,08
12 DATA C5,E5,CD,4A,00,57,E6,18
13 DATA 47,7A,E6,C0,4F,7A,E6,20
14 DATA 57,78,FE,00,28,02,06,20
15 DATA 79,FE,00,28,02,0E,80,7A
16 DATA FE,00,28,02,16,40,78,B1
17 DATA B2,CD,4D,00,E1,C1,23,0B
18 DATA 78,B1,20,CC,C9,FE,02,20
19 DATA 24,CD,FB,C0,CD,1A,C1,E5
20 DATA 09,CD,4A,00,57,79,FE,04
21 DATA 30,06,AF,7A,0F,CD,4D,00
22 DATA E1,03,79,FE,08,20,E8,CD
23 DATA 35,C1,20,E0,C9,FE,03,20
24 DATA 44,CD,0B,C1,CD,1A,C1,E5
25 DATA 09,CD,4A,00,57,79,FE,01
26 DATA 30,07,AF,7A,0F,0F,0F,18
27 DATA 1C,FE,02,30,07,AF,7A,0F
28 DATA 0F,0F,18,11,FE,03,30,06
29 DATA AF,7A,0F,0F,18,07,FE,04
30 DATA 30,06,AF,7A,0F,CD,4D,00
31 DATA E1,03,79,FE,08,20,C8,CD
32 DATA 35,C1,20,C0,C9,FE,04,20
33 DATA 24,CD,FB,C0,CD,1A,C1,E5
34 DATA 09,CD,4A,00,57,AF,7A,0F
35 DATA 57,79,FE,04,7A,30,03,AF
36 DATA 7A,07,CD,2A,C1,20,E8,CD
37 DATA 35,C1,20,E0,C9,FE,05,C0
```

```
38 DATA CD,0B,C1,CD,1A,C1,E5,09
39 DATA CD,4A,00,57,AF,7A,0F,B2
40 DATA CD,2A,C1,20,F1,CD,35,C1
41 DATA 20,E9,C9,CD,6C,00,3A,B7
42 DATA F3,6F,23,3A,B8,F3,67,01
43 DATA 00,00,C9,CD,6F,00,3A,C1
44 DATA F3,6F,3A,C2,F3,67,01,00
45 DATA 00,C9,D1,C5,E5,D5,09,09
46 DATA 09,09,09,09,09,09,01,00
47 DATA 00,C9,CD,4D,00,D1,E1,D5
48 DATA 03,79,FE,08,C9,D1,E1,C1
49 DATA D5,03,78,FE,01,C9,C9,58
50 DATA 53,57,C9,C9,C9,C9,C9,C9
100 REM ********************
110 REM ** LE LINHAS DATA **
120 REM ********************
130 SCREEN 0 : WIDTH 40 : KEY OFF
140 SCREEN 1 : WIDTH 32
150 DEFUSR=&HC000 : DEFINT A-Z
160  FOR F=&HC000 TO &HC141 STEP 8
170   SOMA=0
180    FOR K=0 TO 7
190     N=(F-&HC000)/8+10
200     READ A$:POKE F+K,VAL("&H"+A$)
210     SOMA=SOMA+VAL("&H"+A$)
220    NEXT K
230   PRINTUSING"##";N;
240   PRINT"=";
250   PRINTUSING"####";SOMA;
260   PRINT" ";
270  NEXT F
300 REM **************************
310 REM ** ROTINA DE CONFERENCIA **
320 REM **************************
330 PRINT:PRINT"CONFERE?(S/N)"
340 A$=INPUT$(1)
350 IF A$="N" OR A$="n" THEN PRINT
   "EM QUE LINHA HA ERRO"; ELSE 500
360 INPUT L
370 IF L>50 OR L<10 THEN GOTO 360
380 LOCATE 8,17:PRINT"APERTE RETURN!"
390 LOCATE 1,18:PRINT"LIST ";L
400 LOCATE 0,16:STOP
500 REM ****************************
510 REM ** ROTINA DE DEMONSTRACAO **
520 REM ****************************
530  FOR G=1 TO 5
540   POKE0,USR0(G)
```

```
550     FOR F=ASC("0") TO ASC("z")
560       PRINT CHR$(F);" ";
570     NEXT F
580    IF STRIG(0)=0 THEN 580
590   NEXT G
600 SCREEN 0
```

Ao rodar o programa , você deve obter na tela os códigos listados na figura 27.2

Figura.27.2-Códigos corretos

```
10= 899 11= 748 12=1046 13=1078
14= 541 15= 681 16= 679 17= 924
18=1022 19=1337 20= 754 21= 648
22=1080 23= 992 24=1130 25= 751
26= 421 27= 651 28= 382 29= 616
30= 648 31=1048 32= 961 33=1337
34= 687 35= 814 36=1038 37=1154
38=1071 39= 856 40=1164 41=1020
42= 978 43= 768 44= 953 45=1067
46=  55 47=1130 48=1214 49=1081
50=1376
CONFERE?(S/N)
█
```

Confira todos os códigos da tela com os da figura 27.2. Se algum não conferir, tecle N (de Não) para poder afetuar correção conforme explicado a seguir.

Digamos, por exemplo, que você tenha obtido a tela de figura 27.3

Figura. 27.3

```
10= 899 11= 748 12=1046 13=1078
14= 541 15= 681 16= 682 17= 924
18=1022 19=1337 20= 754 21= 648
22=1080 23= 992 24=1130 25= 751
26= 421 27= 651 28= 382 29= 616
30= 648 31=1048 32= 961 33=1337
34= 687 35= 814 36=1038 37=1154
38=1071 39= 856 40=1164 41=1020
42= 978 43= 768 44= 953 45=1067
46=  55 47=1130 48=1214 49=1081
50=1376
CONFERE?(S/N)
EM QUE LINHA HA ERRO? 16


Break in 400
Ok      APERTE RETURN!
█LIST  16
```

Checando, você percebe que, na 2ª linha, ao invés de 16= 679 você obteve 16=682.

NÃO CONFERE? (S/N)

você tecla "N" e surge a pergunta.

EM QUE LINHA HÁ ERRO?

você teclará 16 (e RETURN) e aparecerá a mensagem:

APERTE RETURN! com o cursor sobre a linha:

```
LIST 16
```

Obdientemente, você aperta o RETURN e a linha em questão é listada na tela (figura 27.4)

Figura. 27.4

```
10=  899 11=  748 12=1046 13=1078
14=  541 15=  681 16=  682 17=  924
18=1022 19=1337 20=  754 21=  648
22=1080 23=  992 24=1130 25=  751
26=  421 27=  651 28=  382 29=  616
30=  648 31=1048 32=  961 33=1337
34=  687 35=  814 36=1038 37=1154
38=1071 39=  856 40=1164 41=1020
42=  978 43=  768 44=  953 45=1067
46=   55 47=1130 48=1214 49=1081
50=1376
CONFERE?(S/N)
EM QUE LINHA HA ERRO? 16


Break in 400
Ok        APERTE RETURN!
 LIST  16
16 DATA FE,00,2█,02,16,40,78,B1
Ok
```

Checando os códigos com o programa original você percebe que, no 3º código, ao invés de 28 você digitou 2B. Leve o cursor até o B, com as setas, e digite: 8 e RETURN. Agora é só dar RUN e, se necessário, repetir o processo até eliminar todos os erros.

Quando, finalmente, você responder "S", o programa, com muita rapidez, redefine os caracteres (estreitos) e lista alguns na tela. Aperte a barra de espaços e surgirá a segunda fonte (itálico "côncavo") Apertando outra vez a barra você obtem um itálico na SCREEN 1, na próxima outro itálico inclinado para esquerda e, finalmente, um bold.

Aí, o programa em BASIC termina e você recebe o cursor de volta.

Se quiser chamar alguma fonte, comande:

```
G%= número da fonte (de 1 a 5)
POKE 0, USR0 (G%)
```

Agora você pode gravar seu programa em Linguagem Máquina diretamente em código binário. Digite:

```
BSAVE " GERTABS. BIN", &HC000, &HC141
```

Para recuperar este programa, basta digitar o programa da figura 27.5.

Figura. 27.5

```
100 DEFUSR=&HC000
110 BLOAD"GERTABS.BIN",R
120 INPUT"QUAL TABELA(1/5)";G%
130 IF G%<1 OR G%>5 THEN 120
140 POKE0,USR0(G%)
150 LIST
```

Para você se orientar na escolha da tabela, veja a figura 27.6

Figura.27.6- Tabelas geradas (de 1 a 5)

```
0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C
D E F G H I J K L M N O P Q R S T U V W
TABELA  1

0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C
D E F G H I J K L M N O P Q R S T U V W
TABELA  2

0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
TABELA  3

0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C
D E F G H I J K L M N O P Q R S T U V W
TABELA  4

0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
TABELA  5
```

ENTRANDO COM FóRMULAS NO INPUT

28

Imagine uma situação na qual o usuário de um programa deva, além de fornecer os dados numéricos a serem introduzidos numa fórmula, digitar a própria fórmula!

No saudoso BASIC-TK (Sinclair) isso não representava o menor problema, pois seu INPUT admite tanto números quanto expressões.

Já no BASIC MSX, o INPUT admite apenas números ou strings.

A primeira solução que ocorre é entrar com uma string que contenha a fórmula e depois calcular seu valor com a função VAL. Mais uma vez não vai funcionar no MSX (no Sinclair funcionaria!) pois os caracteres que não representem algarismos são rejeitados pelo VAL do BASIC MSX.

Não podemos brecar o programa, introduzir fórmula numa linha do BASIC e continuar rodando, pois toda alteração no programa em BASIC limpa a área de variáveis, fazendo com que percamos os valores introduzidos e calculados até então.

Uma solução que nos foi sugerida pelo Dr. Paulo Saraiva do HC de São Paulo, foi de brecar o programa com o comando STOP, introduzir a fórmula no modo direto e a seguir comandar CONT.

Veja o programa da figura 28.1.

Figura 28.1

```
100 CLS
110 LOCATE 0,20:INPUT "X=";X
120 LOCATE 0,21:INPUT "Y=";Y
130 INPUT "FORMULA Z=";A$
140 LOCATE 15,0:PRINT"aperte RETURN 2x"
150 LOCATE 0,2:PRINT"Z=";A$
160 LOCATE 0,4:PRINT"CONT
170 LOCATE 0,0:STOP
180 LOCATE 0,22:PRINT
190 PRINT "Z=";A$;"=";Z
200 PRINT:PRINT:PRINT
210 PRINT"P/CONTINUAR:BARRA DE ESPACOS"
220 IF STRIG(0) GOTO 100 ELSE GOTO 220
```

As linhas de 110 a 130 pedem os valores de X . Y

e a fórmula (que não deve conter erros de sintaxe e póde ser função de X e Y).

Experimente, por exemplo, digitar X=3, Y=4 e, na fórmula, Z=X*Y^2.

Você obterá o tela da figura 28.2

Figura 28.2

```
Parei em 170   aperte RETURN 2x
Ok
Z=X*Y^2

CONT







X=? 3
Y=? 4
FORMULA Z=? X*Y^2
```

Após a entrada dos dados e fórmula, o programa imprime o comando direto de atribuição, na linha 2 do vídeo, o comando direto de CONT na linha 4 e dá um STOP (veja linha 170) 2 linhas acima (isso é importante!).

Com isso o BASIC imprime a mensagem "Break in 170" e faz o cursor cair exatamente em cima da linha de atribuição!

Digitando RETURN uma vez, ela é executada como comando direto e o cursor cai em cima da linha onde está impresso o CONT. Digitando RETURN mais uma vez o programa continua sua execução imprimindo o resultado na parte inferior da tela e fazendo desaparecer o "pasticcio" na parte de cima (fig 28.3).

Figura 28.3

```
X=? 3
Y=? 4
FORMULA Z=? X*Y^2
Z=X*Y^2= 48
P/CONTINUAR:BARRA DE ESPACOS
```

Para que você possa implementar esse truque em seus próprios programas, lembre-se então:

1- A fórmula entra com o INPUT de uma variável STRING (A$ no nosso exemplo).

2- O programa deve imprimir um comando de atribuição do tipo:

```
PRINT"Z=";A$
```

em uma linha determinada da tela .

3- Posicionar o cursor, com o comando LOCATE duas linhas acima da de atribuição e usar o comando STOP.

4- Feito isso, basta que o usuário digite RETURN duas vezes (coloque uma mensagem lembrando-o disso, como fizemos no programa exemplo) e o problema está resolvido!

Obs: Se você quiser fazer uma rotina mais "idiot proof", veja a próxima dica.

29 COLOCANDO UM "FANTASMA DIGITANDO"

Antes de ler esta dica, é conveniente consultar a dica anterior para se inteirar do problema que queremos resolver.

Basicamente o problema é evitar que o usuário fique digitando RETURN duas vezes para que a uma variável seja atribuído o valor de uma fórmula decidida pelo próprio usuário.

O ideal é colocar esses dois RETURNs no BUFFER do teclado de maneira a simular sua digitação no instante em que o cursor é liberado pelo STOP.

Desta forma, é como se um "fantasma" assumisse o controle do teclado quando o programa é interrompido, não deixando o usuário fazer besteiras!

Para isso, porém, é conveniente saber como funciona o BUFFER do teclado.

Existe uma região de 40 bytes na RAM, que começa no endereço &HFBF0 (64496 em decimal) onde o que está sendo digitado é armazenado; e, dependendo da disponibilidade do micro, é esvaziado.

Quando o micro é liberado para digitação (no modo direto ou através de comandos como o INPUT) ele lê o que está no BUFFER e vai executando como se tivesse sido digitado naquele momento. Se o BUFFER estiver vazio, o micro fica esperando a pressão de uma tecla.

O micro, porém, precisa saber a todo instante em que posição do BUFFER ele deve colocar o próximo byte fornecido pela leitura do teclado.

O endereço dessa posição fica armazenado numa variável chamada PUTPNT que esta nos endereços &HF3F8 e &HF3F9.

Como você deve saber, qualquer endereço da RAM de 64 Kbytes do MSX pode ser armazenado em dois bytes, um chamado de menos significativo (LSB) e outro de mais significativo (MSB).

O endereço é dado pela relação:

End=LSB+256*MSB

Para descobrir, por exemplo, para que endereço do BUFFER está apontando a variável PUTPNT neste

instante, rode o programa da figura 29.1.

figura 29.1

```
100 LP=PEEK(&HF3F8)
110 MP=PEEK(&HF3F9)
120 EP=LP+256*MP
130 PRINT"PUTPNT APONTA P/O ENDERECO:"
140 PRINT EP;"=";
150 PRINT "&H";HEX$(EP)
160 PRINT "DO BUFFER DE TECLADO"
```

Rode o programa apertando F5 (se você possuir um HOTBIT pressione F10). A tela deve mostrar o endereço do BUFFER apontado por PUTPNT em decimal e hexadecimal. Aperte F5 (ou F10) várias vezes seguidas. A cada pressionamento da tecla de função você estará inserindo 4 bytes no BUFFER do teclado (3 do "run" e um do RETURN=CHR$(13)).

Como você pode notar, o endereço apontado por PUTPNT é incrementado de 4 em 4 unidades a cada pressionamento da tecla de função.

Num dado momento você deve ter notado que o endereço cai bruscamente para um valor próximo de 64496 e depois volta a crescer. Isso acontece porque o BUFFER tem uma lógica "circular": quando seu quadragésimo byte é preenchido, ele volta para o primeiro.

Dado esse aspecto "circular" do BUFFER o primeiro byte que o micro retirará dele, assim que se liberar de outras tarefas, não está necessariamente no começo do BUFFER (&HFBF0). Aliás o BUFFER não tem começo, ele é circular!

Assim sendo, existe uma variável chamada GETPNT (LSB em &HF3FA e MSB em &HF3FB) cuja função é justamente de indicar ao micro qual o endereço do primeiro byte a ser "colhido" do BUFFER.

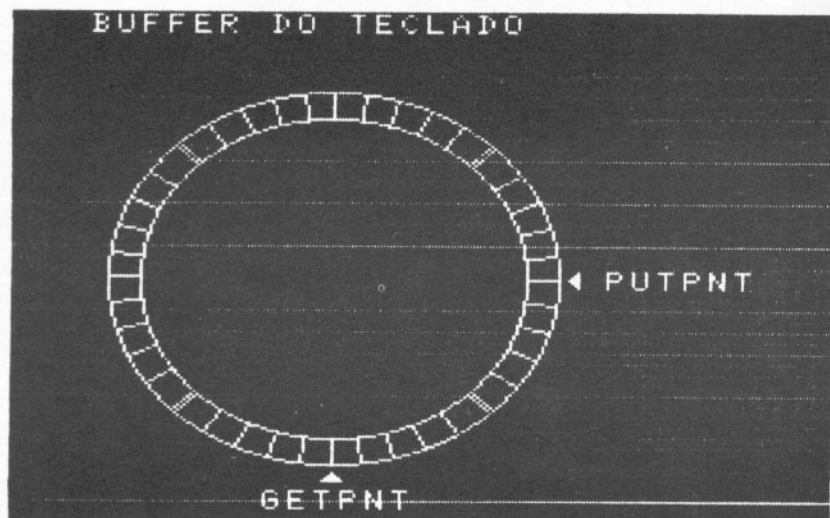
Note que, conforme o buffer vai sendo lido pelo micro liberado, o GETPNT vai "andando" ao longo do BUFFER de um em um byte. Os bytes lidos, porém, não são retirados: isso faz com que os 40 bytes fiquem normalmente cheios de "sujeira".

Obviamente, para que a "sujeira" não seja lida, o micro só lê bytes do "BUFFER" enquanto o GETPNT não alcançar o PUTPNT (fig 29.2)

Imagine então BUFFER como uma pista de corridas onde o GETPNT fica correndo atrás do PUTPNT.

Quando o micro está ocupado com alguma tarefa e

você está digitando no teclado, o PUTPNT vai andando (e você ouve o "click" do teclado). Se, nessa corrida, o PUTPNT alcançar o GETPNT, não o ultrapassa (pois nesse caso perderíamos informções já digitadas) mas pára e você deixa de ouvir o "click": o BUFFER está cheio!

figura 29.2

Quando o micro é liberado, quem começa a "correr" é o GETPNT que também como já vimos, não ultrapassa seu "adversário de corrida" sob pena de começar a ler sujeira.

Existe uma rotina no BIOS do MSX que limpa o BUFFER do teclado (veja "100 DICAS para MSX, programa 1.2, página 9) e que pode ser chamada a qualquer momento por

```
DEFUSR=&H156:POKE 0,USR(0)
```

Na realidade ela não "limpa" nada: o BUFFER continua com a sujeira anterior!

Ela apenas iguala os endereços apontados por GETPNT e PUTPNT simulando,para o micro a situação "JÁ LI TUDO!".

Digite, agora, o programa da figura 29.3.

figura 29.3

```
100 FOR I=62456! TO 62459!
110 POKE I,251
120 NEXT I
130 POKE 62458!,252
```

e não o rode ainda! Apenas liste-o na tela com F4

Agora aperte F5 (ou F10 no HOTBIT). Provavelmente o micro vai "desembestar" e só vai parar se você digitar insistentemente CONTROL+STOP .

O que aconteceu? Ora, nas linhas 100 a 120 escrevemos o mesmo endereço em GETPNT e PUTPNT:

251+256*251=64507

simulando a situação de BUFFER vazio.

Em 130 somamos 1 ao endereço apontado pelo GETPNT: isso equivale a atrasar o PUTPNT em 39 posições, simulando BUFFER cheio. Assim que o micro é liberado, ele lê do BUFFER o LIST e o RUN (que estão como sujeira do BUFFER) e os executa entrando num circulo vicioso!

Vejamos agora como dar uma utilidade a tudo que aprendemos, resolvendo o problema abordado na dica anterior.

Digite o programa da figura 29.4

figura 29.4

```
100 SCREEN0:RESTORE
110 INPUT"X=";X:INPUT"Y=";Y
130 INPUT"FORMULA-> Z=";Z$
140 LOCATE 0,10:PRINT"Z=";Z$
150 FOR I=1 TO 6
160 READ A$,B$
165 AH=VAL("&H"+A$):BH=VAL("&H"+B$)
170 POKE AH,BH
180 NEXT I
190 DATAFBF0,0D,FBF1,0D,F3F8,F2,F3F9,FB
195 DATA F3FA,F0,F3FB,FB
200 LOCATE 0,12:PRINT"CONT"
210 LOCATE 0,8:STOP
220 CLS:PRINT"X=";X:PRINT"Y=";Y
240 PRINT:PRINT"Z=";Z$;"=";Z
250 PRINT:PRINT"OUTRA VEZ ? (S/N)"
260 T$=INPUT$(1):IF T$="S" OR T$="s" THE
```

```
N 100 ELSE LIST
```

As linhas de 100 a 130, limpam a tela e dão entrada aos valores de X, Y e fórmula.

A linha 130 localiza a atribuição de Z, no modo direto, na posição 0,10 da tela.

Obviamente o CONT deve estar na posição 0,12 (veja linha 200) e o cursor deve estar na posição 0,8 quando for comandado o STOP (veja linha 210).

A novidade está entre as linhas 150 e 195: Os endereços que estão nas linhas DATA e os POKEs da linha 170 fazem o seguinte:

As instruções:

```
POKE &HFBF0,&H0D
POKE &HFBF1,&H0D
```

colocam dois RETURNs (CHR$(&H0D)=RETURN) no BUFFER a partir da posição &HFBF0.

As instruções:

```
POKE &HF3F8,&HF2
POKE &HFBF9,&HFB
```

fazem o PUTPNT apontar para o endereço HFBF2 do BUFFER, ou seja, imediatamente depois dos dois RETURNs (que estão em &HFBF0 e &HFBF1).

As instruções:

```
POKE &HF3FA,&HF0
POKE &HF3FB,&HFB
```

fazem o GETPNT apontar para o endereço &HFBF0, ou seja, onde está o primeiro RETURN a ser colhido pelo BUFFER do teclado.

Quando o micro é liberado pelo STOP, o cursor cai sobre a linha de atribuição e o primeiro RETURN é lido e executado.

O valor de Z$ é atribuido a Z e o cursor cai sobre o CONT, liberando novamente o micro.

Ele lê e executa o segundo RETURN fazendo que o programa continue a sua execução normalmente.

Como você pode notar, o programa ficou bastante "idiot proof" pois no momento em que a execução do programa é interrompida pelo STOP não há necessidade de fazer o usuário digitar nada: o "fantasma" que você instalou no BUFFER se encarrega disso!

Para ver uma outra interessante aplicação desse "fantasma", leia a próxima dica.

LINHA DATA AUTOMÁTICA

30

Vimos na dica anterior (29) o funcionamento do buffer do teclado, e como simular a digitação de caracteres.

O "fantasma" digitava o valor das fórmulas no modo direto, ou seja não havia numeração nos comandos.

Você conseguiria imaginar o que aconteceria se houvesse um número antes do comando?

Quando o comando STOP é executado, o programa pára, e o micro retorna ao modo direto, onde o programa pode ser editado. Se imprimirmos uma linha de programa na tela para que o "fantasma" digite ela será incluída no programa.

Porém, existe um grande incoveniente em fazer esse truque, pois ao inserirmos uma linha de programa perdemos o conteúdo de todas as variáveis.

Quando se trata de programas que possuem poucas variáveis, esse problema pode ser amenizado.

É uma idéia muito simples. Basta guardar essas variáveis na memoria RAM com o comando POKE. Para recuperá-las basta usar o comando PEEK.

O programa da figura 30.1 é uma aplicação útil usando esse tipo de truque. Ele imprime o número da linha e o comando DATA no vídeo e executa o truque do "fantasma", mas antes (linhas 130 e 140) guarda os valores inicial e final dos números das linhas em uma região livre da RAM.

Na linha 190 o número da linha a ser gerada é recuperado da memória e impresso com o comando na linha 200.

A linha 210 testa se o número da linha gerada é maior que o da última linha. Caso afirmativo a tela é apagada, e ao ser dado o comando STOP o fantasma digita o RETURN em linhas em branco e o programa pára de operar.

Caso ainda hajam linhas a serem geradas variável I (que fornece o número da linha) é armazenada na memória.

As linhas de 220 a 240 atualizam as variáveis PUTPNT e GETPNT para simular a digitação da tecla RETURN.

O comando LOCATE da linha 250 posiciona o cursor no lugar correto.

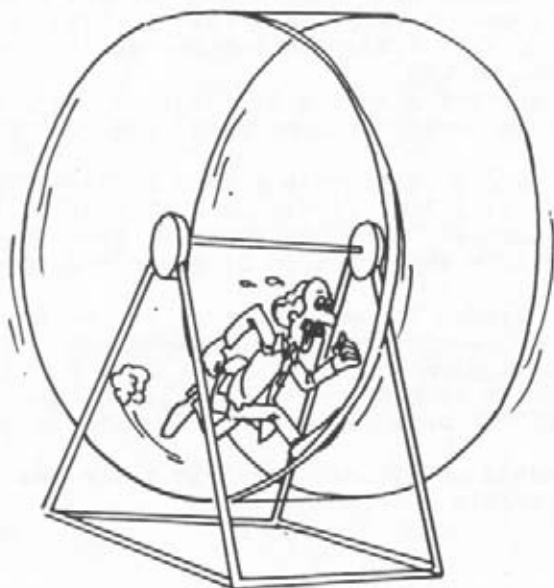
Figura 30.1

```
100 CLS
110 INPUT"linha inicial (>1000)";LI
120 INPUT"linha final (<3540)";LF
130 POKE &HFFCA,(LI-1000)/10
140 POKE &HFFCB,(LF-1000)/10
150 POKE &HFBF0,13
160 POKE &HFBF1,13
170 LOCATE 0,11:PRINT"goto 80
180 LOCATE 0,10
190 I=1000+PEEK(&HFFCA)*10
200 PRINT I;"DATA XX,XX,XX,XX,XX,XX,XX,X
X"
210 IF PEEK (&HFFCA)+1>PEEK(&HFFCB) THEN
 CLS ELSE :POKE &HF3F8,&HF2:I=I+10:POKE
&HFFCA,(I-1000)/10
220 POKE &HF3F9,&HFB
230 POKE &HF3FA,&HF0
240 POKE &HF3FB,&HFB
250 LOCATE 0,8
260 STOP
```

Esse programa pode auxiliá-lo na digitação de programas BASIC que possuam muitas linhas DATA.

Altere a linha 200 para outros comandos do BASIC e veja seu efeito.

SIMULANDO A TECLA ALT DO PC

31

A tabela de caracteres dos MSX dispõe de 256 caracteres, o que é de grande utilidade, pois além da tabela ASCII que é padrão na grande maioria dos micros, temos caracteres acentuados, gráficos, simbolos matemáticos e até letras de outros alfabetos (grego, holandês, etc).

A grande dificuldade está em achar um desses caracteres "especiais", pois temos 49 teclas do teclado QWERTY para os 256 caracteres. É meio trabalhoso ficar decorando que teclas apertar (RGRA, LGRA,SHIFT, GRAPH,CODE) para que uma simples barrinha para moldura ou uma "carinha" apareçam no vídeo.

Nos computadores da linha PC esse problema foi resolvido com uma idéia muito interessante. Existe uma tecla (chamada ALT) que quando pressionada permite que seja digitado o código do caracter. Afinal, é muito mais fácil achar o código do caracter em uma tabela de caracteres, do que achar a posição de uma tecla em várias tabelas.

Devido à grande versatilidade dos MSX, resolvemos "chupar" (e com sucesso!) essa característica do PC. Afinal, as boas idéias são feitas para serem aproveitadas (principalmente se facilitarem a vida do usuário).

Para transformarmos essa idéia em um programa que funcione, precisamos das seguintes informações:

Como interceptar a varredura do teclado.
Ler uma tecla pressionada.
Saber a localização do buffer do teclado.

Interceptar a varredura do teclado é uma tarefa muito simples. Existe uma hook (HKEYC) no endereço &HFDCC que é chamada sempre que uma tecla é pressionada. O valor para se obter o código da tecla está no registrador A e no registrador C.

Usaremos a tecla SELECT como "ALT", (isso pode ser alterado) e o seu código quando interceptado na hook HKEYC é &H3E.

Mudando a hook HKEYC para o inicio do programa teremos a seguinte condição:

É o código &H3E? (tecla SELECT)

Caso afirmativo, leremos(usando a rotina CHGET do BIOS) o código de duas teclas pressionadas (2 dígitos hexadecimais). Em seguida calculamos o código do caracter.

Caso a tecla pressionada não seja a SELECT, o programa não altera nada e devolve o controle do micro para a ROM.

Em posse do código do caracter, basta colocá-lo no buffer do teclado.

Optamos, nesse programa por colocar o caracter no inicio físico do buffer do teclado (&HFBF0), e apontar as variáveis de sistema PUTPNT (&HF3F8) e GETPNT (&HF3FA) para o endereço &HFBF1 e &HFBF0 respectivamente.

A listagem a seguir é uma versão BASIC que instala o programa. A listagem do programa em Assembler encontra-se no Apendice I no final do livro.

```
1000 DATA 21,0C,D0,22,CD,FD,3E,CD
1010 DATA 32,CC,FD,C9,FE,3E,C0,E5
1020 DATA C5,D5,F5,CD,56,01,CD,9F
1030 DATA 00,FE,3A,38,02,D6,07,D6
1040 DATA 30,CB,27,CB,27,CB,27,CB
1050 DATA 27,E6,F0,57,CD,9F,00,FE
1060 DATA 3A,38,02,D6,07,D6,30,B2
1070 DATA 21,F0,FB,22,FA,F3,FE,20
1080 DATA 38,07,77,23,22,F8,F3,18
1090 DATA 0D,57,3E,01,77,23,3E,40
1100 DATA 82,77,23,22,F8,F3,F1,D1
1110 DATA C1,E1,C9
1120 FOR L=&HD000 TO &HD05A
1130 READ A$
1140 POKE L,VAL("&H"+A$)
1150 NEXT
1160 DEFUSR=&HD000
1170 POKE 0,USR(0)
1180 END
```

NUMÉRICO HEXADECIMAL

32

Quem adquiriu o livro "CEM DICAS PARA MSX" e digitou a maioria dos programas, deve ter se tornado um exímio digitador de códigos hexadecimais (se ainda não foi internado em um manicômio!).

O teclado numérico reduzido do Expert ajuda a digitação de números, mas quando são códigos hexadecimais, a coisa fica meio chata. Afinal, ter que procurar as letras no teclado QWERTY "quebra", em muito, o ritmo da digitação.

Vamos analisar a disposição do numérico reduzido do Expert (figura 32.1)

Figura 32.1

As teclas de operação não são as mesmas do teclado QWERTY. Elas são outros elementos da matriz do teclado (veja no APROFUNDANDO-SE NO MSX pág 84).

As teclas de numeros, de ponto decimal e do sinal de igual são teclas em "paralelo" com as do teclado QWERTY. Você pode observar isso pelas seguinte características:

Pressionando-se a tecla de igual do teclado QWERTY até que o auto-repeat começe a operar e em seguida pressionando-se a tecla de igual do numérico reduzido, o auto-repeat não pára de operar, e nem tão pouco começa a imprimir o dobro de "iguais" como faria se duas teclas distintas fossem pressionadas simultaneamente.

Além disso, as teclas de SHIFT funcionam com essas teclas. Experimente ficar pressionando SHIFT e apertar algumas teclas do numérico. Você deverá obter

os caracteres que ficam acima das teclas de números do teclado QWERTY.

Quando interceptamos a hook HKEYC (&HFDCC) obtemos no registro A um código que representa uma tecla pressionada. O registro C também possui o mesmo código e quando o ele é de alguma tecla do teclado QWERTY precisamos alterar o registrador C para o mesmo valor.

Veja na figura 32.2 quais os códigos que correspondem às teclas que nos interessam.

Figura 32.2

TECLA	CÓDIGO
/	4B
*	4A
-	49
+	48
=	0B
.	13
SHIFT	30

TECLA	CÓDIGO
F	1B
E	1A
D	19
C	18
B	17
A	16

Para que o novo numérico reduzido possua a distribuição da figura 32.3, podemos usar a seguinte fórmula para o cálculo do código da nova tecla.

novo código= 61-código atual

Essa fórmula não vale para o ponto decimal nem para o sinal de igual, pois como elas se situam em outra parte da matriz do teclado, o seu código é totalmente diferente. A solução mais fácil, então, é colocar duas condições no programa para detectar essas teclas e converter o seu código para o das letras "A" e "B".

Figura 32.3

Para que o " numérico - hexa " não fique o tempo todo operando e impeça que você o use para contas em decimal, usaremos um "liga-desliga" que será feito pela tecla SHIFT (código &H30), pressionada conjuntamente com alguma tecla de operação do numérico reduzido (/,*,- ou +).

Com a hook HKEYC (&HFDCC) desviada para o programa, faremos as seguintes condições com o registro A:

É o código &H30? (tecla SHIFT). Caso afirmativo, o programa liga uma flag; pois se próxima tecla for um sinal de operação o programa liga ou desliga o numérico hexadecimal.

O código é menor que &H48? (teclas /,*,- ou +) **Caso negativo** o programa realiza as seguinte operações:

Desliga a flag do SHIFT.

Verifica se o numérico está ligado. **Caso afirmativo**, testa as teclas de ponto decimal e do sinal de igual. **Caso negativo** retorna para a ROM (pois a tecla pressionada não nos interessa).

Caso as teclas de operação tenham sido pressionadas, o programa realiza as contas para converter o código e o substitui pelo valor antigo no registro A.

Se você conhece Linguagem de Máquina, dê uma olhada na listagem do programa fonte no Apêndice I.

```
1000 DATA 21,0C,D0,22,CD,FD,3E,C3
1010 DATA 32,CC,FD,C9,FE,30,28,55
1020 DATA FE,48,38,1B,D5,C5,E5,F5
1030 DATA 3A,6D,D0,FE,FF,28,35,3A
```

```
1040 DATA 6E,D0,FE,00,28,2B,F1,5F
1050 DATA 3E,61,93,E1,C1,D1,C9,F5
1060 DATA 3E,00,32,6D,D0,3A,6E,D0
1070 DATA FE,00,28,0A,F1,F5,FE,0B
1080 DATA 28,06,FE,13,28,07,F1,C9
1090 DATA F1,AF,0E,17,C9,F1,0E,16
1100 DATA C9,F1,18,D7,3E,00,32,6D
1110 DATA D0,3A,6E,D0,2F,32,6E,D0
1120 DATA F1,D1,C1,E1,C9,F5,3E,FF
1130 DATA 32,6D,D0,F1,C9,00,00,64
1140 DATA 6F
1150 FOR L=&HD000 TO &HD06F
1160 READ A$
1170 POKE L,VAL("&H"+A$)
1180 NEXT
1190 DEFUSR=&HD000
1200 POKE 0,USR(0)
1210 END
```

A HOOK HKEYC

33

Existem vários programas em linguagem de máquina que alteram a hook HKEYC para detectar se uma determinada tecla foi pressionada (ALT-dica 31, cópia gráfica- 100 dicas).

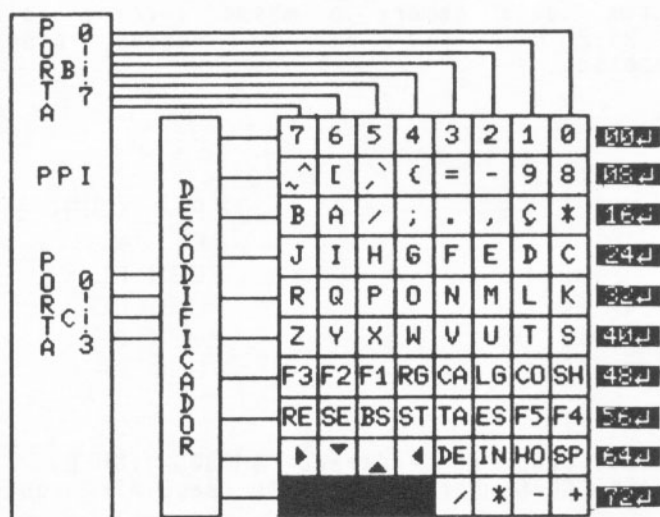
Porém, muitas vezes, a tecla que o programa intercepta para entrar em funcionamento não é a ideal para algum software que estamos elaborando.

Procurar em qual endereço essa procura é feita, não é muito difícil, mesmo para quem não conhece linguagem de máquina.

Para elaborar essa mudança, precisamos saber qual o código da tecla que o programa testa, e qual o novo código para a qual vamos mudar o programa.

Quando a hook HKEYC é interceptada, os registros A e C contém um código, obtido da matriz do teclado que nada tem a ver com o código ASCII do caracter.

Calcular esse código é muito fácil. Obseve na figura 33.1 a matriz do teclado. Basta começar a contar, de cima para baixo e da direita para a esquerda até chegar ao elemento da matriz que representa a tecla.

Figura 33.1

Execute o programa ALT (dica 31) e vamos mudar o "ALT" da tecla SELECT para a tecla ESC.

O código para as duas teclas é o seguinte

```
SELECT    &H3E (62 em decimal)
ESC       &H3A (58 em decimal)
```

O hook HKEYC está nos cinco bytes a partir do endereço &HFDCC.

Consultando o conteúdo desses bytes comandando:

```
PRINT HEX$(PEEK(&HFDCC));HEX$(PEEK(&HFDCD
));HEX$(PEEK(&HFDCE))
```

obteremos os seguintes números:

```
CD 0C D0
```

O &HCD é (em linguagem de máquina) um CALL, que é parecido com o comando GOSUB do BASIC.

Os dois numeros seguintes indicam na forma LSB e MSB o endereço para o qual a execução é desviada sempre que uma tecla é pressionada. Ou seja, após rodar o programa ALT, a hook HKEYC foi desviada para o endereço &HD00C.

Tendo uma vez esse endereço, basta consultar a memória a partir dele, e com um pouco de sorte acertaremos a mudança da tecla.

A maioria dos programas que alteram a hook HKEYC apresentam, quase sempre o mesmo início. Veja na figura 33.2 o início de um desses programas disassemblado.

Figura 33.2

```
D00C           INICIO:
D00C FE3E            CP   03EH
D00E C0              RET  NZ
D00F E5              PUSH HL
D010 C5              PUSH BC
D011 D5              PUSH DE
D012 F5              PUSH AF
```

A instrução no endereço &HD00C (&HFE) é uma comparação (CP=ComPare) e o byte seguinte contém o

código da tecla (que no caso é o código para a tecla SELECT=&H3A).

Basta, então, comandar:

```
POKE &HD00C, novo código
```

testar para ver se a alteração está correta e gravar a nova versão do programa.

O programa BASIC da figura 33.2 seguir faz essa procura automaticamente. Pergunta qual código procurar, qual o novo código e apresenta o conteúdo dos bytes na tela até encontrar a seqüencia desejada.

Podem existir outros programas que usem outras instruções para fazer a comparação da tecla. Nesse caso, a alteração se torna muito mais difícil, pois existem inúmeras possibilidades.

Figura 33.3

```
100 IF PEEK(&HFDCC)=&HC9 THENPRINT"HOOK N
ÃO ALTERADA!":BEEP:END
110 INPUT"QUAL CóDIGO A SER PROCURADO";CD
120 INPUT"QUAL NOVO CóDIGO";NC
130 ED=PEEK(&HFDCD)+256*PEEK(&HFDCE)
140 PRINT"HOOK ALTERADO PARA O ENDEREÇO &
H"HEX$(ED)
150 FOR L=0 TO 1000
160 PRINT HEX$(ED+L)"..."RIGHT$("0"+HEX$(
PEEK(ED+L)),2);
170 IF PEEK(ED+L)=&HFE AND PEEK(ED+L+1)=C
D THEN PRINT" <--- COMPARAÇÃO DO CóDIGO!!
)":GOTO 190:ELSE PRINT
180 NEXT:END
190 PRINT:PRINT"CONFIRMA ALTERACÃO?"
200 A$=INPUT$(1)
210 IF A$="S" OR A$="s" THEN POKE (ED+L+1
),NC:ELSE NEXT L
220 PRINT"Programa Alterado"
230 PRINT"Execute teste e proceda gravação"
240 END
```

34 GERADOR DE LABIRINTOS NO MSX

Para quem gosta de fazer seus próprios jogos, temos aqui um algoritmo muito interessante de gerar labirintos na tela. O modo de tela escolhido foi o SCREEN 1 por apresentar boa definição dos caracteres e e ao mesmo tempo ser fácil de sofrer modificações pelo comando VPOKE.

O programa começa perguntando o tamanho na horizontal e na vertical que deverá ter o labirinto. O labirinto é gerado imediatamente.

Você pode usar esse programa como rotina de um jogo (tipo perseguição) ou imprimir as telas geradas para brincar de "passatempo".

Veja na figura 34.1 o programa.

Figura 34.1

```
10 SCREEN 0:KEYOFF
20 INPUT "Horizontal (3-15) ";H
30 INPUT "Vertical (3-11) ";V
40 SCREEN 1:WIDTH 32
50 PRINT STRING$(H*2+1,215)
60 A$=CHR$(215)+STRING$(H*2-1,219)+CHR$(
215)
70 FOR I=1 TO V*2-1:PRINT A$:NEXT I
80 PRINT STRING$(H*2+1,215)
90 DIM A%(165):D(0)=1:D(1)=-32:D(2)=-1:D
(3)=32
100 X=6177:VPOKE X,32:PP=1
110 IF VPEEK(X+64)<>219 AND VPEEK(X+2)<>
219 AND VPEEK(X-64)<>219 AND VPEEK(X-2)<
>219 THEN 160
120 Y=INT(4*RND(2)):X1=X+2*D(Y)
130 IF VPEEK(X1)<>219 THEN 120
140 VPOKE X1,32:VPOKE X+D(Y),32
150 A%(PP)=X:PP=PP+1:X=X1:GOTO 110
160 PP=PP-1:IF PP=1 THEN 180
170 X=A%(PP):GOTO 110
180 PRINT"PRONTO ";:I$=INPUT$(1):SCREEN0
190 END
```

As linhas de 50 a 80 montam na tela o "tabuleiro" do labirinto.

A matriz D indica as 4 direções do seguinte modo:

D(0)= 1 (direita)
D(1)= -32 (para cima)
D(2)= -1 (esqierda)
D(3)= 32 (para baixo)

O algorítmo escolhe uma das direções aleatóriamente, mas sempre testa se o "caminho" aberto não vai cair em em um lugar já aberto ou na moldura.Caso isso aconteça, é sorteada outra direção.

Sempre que a direção é alterada o endereço da mudança (X) é guardado na matriz **A%** (no índice PP).

Quando não existe nenhuma direção a seguir A variável PP é decrementada e o endreço anterior dado por **A%**(PP) é atribuido à variável X. A seguir é testado se existe outra direção possível.

Quando o algorítmo começa a funcionar, vários endereços são atribuídos na matriz **A%**, ou seja, os dados na matriz começam a crescer. Mas à medida que o labirinto vai sendo construido as possibilidades de caminho se esgotam. Então os dados da matriz **A%** vão sendo consultados, esvaziando-a (decrementando a variável PP).

Quando a váriavel PP chega novamente a 1, é sinal que todas as possibilidades se esgotaram e o labirinto está pronto.

Apresentamos na figura 34.2 uma versão desse algorítmo para a SCRREN 2. Analize as diferenças entre esse porgrama e o anterior.

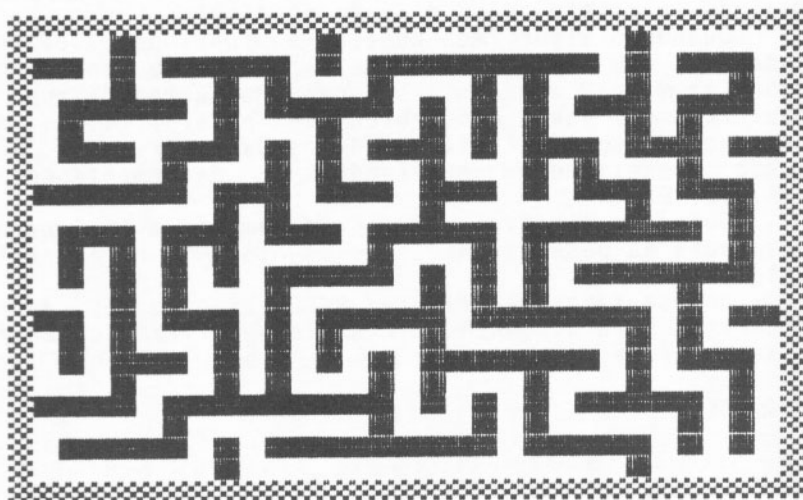
Figura 34.2

```
100 DIM A%(3500,1):DIM DX(4):DIM DY(4)
110 SCREEN 0:KEYOFF
120 INPUT "Horizontal(10-254)";H
130 INPUT "Vertical(10-254)";V
140 SCREEN 2
150 LINE (0,0)-(H,V),15,B
160 DX(0)=1:DX(1)=-1:DX(2)=0:DX(3)=0
170 DY(0)=0:DY(1)=0:DY(2)=1:DY(3)=-1
180 X=2:Y=2:PSET (X,Y):PP=1
190 IF POINT(X,Y+2)=15 AND POINT(X+2,Y)=15
 AND POINT(X,Y-2)=15 AND POINT(X-2,Y)=15
 THEN 270
200 IF X>255 OR X<0 THEN 270
210 IF Y>255 OR Y<0 THEN 270
```

```
220 Z=INT(4*RND(2)):X1=X+2*DX(Z)
230 Y1=Y+2*DY(Z)
240 IF POINT(X1,Y1)=15 THEN 220
250 PSET (X1,Y1),15:PSET(X+DX(Z),Y+DY(Z))
,15
260 A%(PP,0)=X:A%(PP,1)=Y:PP=PP+1:X=X1:Y=
Y1:GOTO 190
270 PP=PP-1:IF PP=1 THEN 290
280 X=A%(PP,0):Y=A%(PP,1):GOTO 190
290 BEEP:GOTO 290
300 END
```

Tente modificar o programa da figura 32.2 para que o algorítmo funcione na SCREEN 3.

PINTANDO FIGURAS COM TEXTURAS

35

Um recurso largamente usado em desenhos por computador é a técnica de reticulado ou textura, ou seja, a tonalidade de um trecho do desenho é dada pela distribuição dos pontos acesos neste trecho. No MSX a técnica de texturas é muito pouco usada, bem menos que em outros computadores mais simples. Este programa vem a ser uma solução para isto.

Existem 11 texturas "pré-fabricadas" neste programa, o que já deve ser mais que suficiente para muitas aplicações.

Uma limitação deste programa é quanto às cores: a memória de cores é usada como rascunho durante a operação do PAINTER, e depois é preenchida com as cores indicadas no comando COLOR. Outro problema acontece se você mandar o programa pintar uma figura aberta: ele não respeita os extremos do vídeo e acaba enchendo a VRAM inteira com dados.

O primeiro programa em BASIC (figura 35.1) gera um arquivo chamado PAINT.BIN, que deve ser carregado com o segundo programa em BASIC.

Para pintar uma área, são necessários três POKEs para indicar a posição e o tipo da textura:

```
POKE &HD032,X
POKE &HD033,Y
POKE &HF975,N
```

X:Coordenada X do ponto inicial.
Y:Coordenada Y do ponto inicial.
N:Número da retícula (0-10).

Figura 35.1

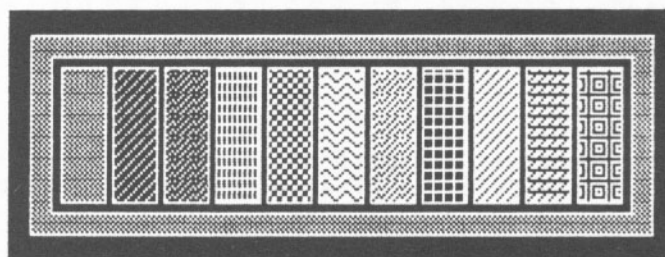
```
1000 DATA 21,00,00,11,00,20,01,00
1010 DATA 18,CD,4A,00,EB,CD,4D,00
1020 DATA EB,13,23,0B,78,B1,20,F1
1030 DATA CD,31,D0,21,00,20,3A,E9
1040 DATA F3,87,87,87,87,47,3A,EA
1050 DATA F3,B0,01,00,18,CD,56,00
1060 DATA C9,11,00,00,21,3F,D1,06
1070 DATA FE,CD,BA,D0,06,01,CD,BA
1080 DATA D0,06,FF,0E,00,CD,03,D1
```

```
1090 DATA A7,28,09,CD,C3,D0,78,FE
1100 DATA FE,C8,18,F1,CD,CA,D0,ED
1110 DATA 53,3D,D1,CB,81,CB,89,15
1120 DATA CD,03,D1,B1,4F,14,14,CD
1130 DATA 03,D1,87,B1,4F,15,3E,01
1140 DATA CB,51,28,02,ED,44,83,5F
1150 DATA CD,03,D1,A7,20,2B,CB,41
1160 DATA 28,0F,15,CD,03,D1,14,A7
1170 DATA 20,07,C5,06,FF,CD,BA,D0
1180 DATA C1,CB,49,28,0F,14,CD,03
1190 DATA D1,15,A7,20,07,C5,06,01
1200 DATA CD,BA,D0,C1,CD,CA,D0,18
1210 DATA B2,ED,5B,3D,D1,CB,51,20
1220 DATA 04,CB,D1,18,A2,7A,80,57
1230 DATA 18,89,73,23,7A,80,77,23
1240 DATA 70,23,C9,2B,46,2B,56,2B
1250 DATA 5E,C9,E5,C5,CD,1A,D1,F5
1260 DATA E5,3A,75,F9,87,87,87,47
1270 DATA 7D,E6,07,B0,06,00,4F,21
1280 DATA 5D,D1,09,4E,E1,F1,E5,C5
1290 DATA 01,00,20,09,C1,47,CD,4A
1300 DATA 00,B0,CD,4D,00,E1,78,A1
1310 DATA 47,CD,4A,00,B0,CD,4D,00
1320 DATA C1,E1,C9,E5,C5,CD,1A,D1
1330 DATA 01,00,20,09,47,CD,4A,00
1340 DATA A0,3E,00,28,02,3E,01,C1
1350 DATA E1,C9,7A,CB,3F,CB,3F,CB
1360 DATA 3F,67,7B,E6,F8,6F,7A,E6
1370 DATA 07,85,6F,3E,00,8C,67,7B
1380 DATA E6,07,47,3E,08,90,47,AF
1390 DATA 37,17,10,FD,C9,BB,40,B3
1400 DATA 44,BF,44,BF,44,BF,44,BF
1410 DATA 44,BF,44,BF,44,BF,44,BF
1420 DATA 44,BF,44,BF,44,BF,44,BF
1430 DATA 44,BF,44,BF,44,AA,55,AA
1440 DATA 55,AA,55,AA,55,88,11,22
1450 DATA 44,88,11,22,44,92,24,49
1460 DATA 92,24,49,92,24,AA,AA,AA
1470 DATA FF,AA,AA,AA,FF,CC,CC,33
1480 DATA 33,CC,CC,33,33,CF,B7,7B
1490 DATA FC,CF,B7,7B,FC,6D,DB,B6
1500 DATA 6D,DB,B6,6D,DB,FF,88,88
1510 DATA 88,FF,88,88,88,77,EE,DD
1520 DATA BB,77,EE,DD,BB,00,EE,DD
1530 DATA BB,00,EE,DD,BB,00,7F,41
1540 DATA 5D,55,5D,41,7F,FIM
1550 CLS:PRINT "CARREGANDO PAINTER"
1560 FOR I=&HD000 TO &HD1B4
```

```
1570 READ A$:POKE I,VAL("&H"+A$)
1580 NEXT I
1590 BSAVE "PAINT.BIN",&HD000,&HD1B4
1600 END
1610 REM
1620 REM ------------------------
1630 REM Programa escrito por
1640 REM  THE PILOT - 1988
1650 REM ------------------------
```

O programa (em BASIC) da figura 35.2 dá uma boa demonstração do PAINTER.

Figura 35.2

```
100 CLEAR 200,&HD000:SCREEN 2
110 BLOAD"PAINT.BIN"
120 DEFUSR=&HD000
130 FOR I=0 TO 10
140 LINE(I*20+20,70)-(I*20+37,120),,B
150 POKE &HD032,I*20+21
160 POKE &HD033,71
170 POKE &HF975,I
180 A=USR(0)
190 NEXT I
200 LINE(16,66)-(241,124),,B
210 LINE(8,58)-(249,132),,B
220 POKE &HD032,9
230 POKE &HD033,59
240 POKE &HF975,0
250 A=USR(0)
260 GOTO 260
```

36 LOCALIZANDO A PAGINAÇÃO DA MEMóRIA

Os 64Kb de memória do MSX são divididos em páginas de 16kb cada.

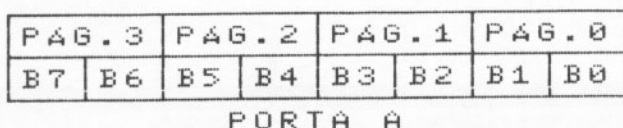
Por outro lado, o MSX possui 4 slots (sendo que cada um pode ser expandido em 4 sub-slots). Isto significa que, apesar do MSX poder trabalhar com apenas 4 páginas de cada vez, cada uma delas pode estar localizada em 16 sub-slots diferentes. Isto permite gerenciar 64x16= 1024 Kb= 1 Megabyte de memória !

Quando trabalhamos em BASIC as páginas 0 e 1 contêm 32 Kb de informações usualmente em memória ROM (na página 0 temos o BIOS e na 1 o interpretador BASIC) e as páginas 2 e 3 contêm 32Kb de memória RAM, sendo que alguns Kbytes são usados pelo próprio sistema para variáveis e/ou eventuais "buffers" do disco.

Para maiores detalhes sobre esta paginação, slot primário e secundário, remetemos o leitor ao "APROFUNDANDO-SE NO MSX", capítulos 0 e 3.

Para que o micro saiba, a cada instante, em que slot está cada página, a porta A da PPI (&HA8) contêm estas informações em 8 bits (figura.36.1).

Figura.36.1

PAG.3		PAG.2		PAG.1		PAG.0	
B7	B6	B5	B4	B3	B2	B1	B0

PORTA A

Para descrever o número do slot bastam 2 bits para cada porta pois, em binário, podemos escrever um número de 0 a 3 com apenas dois dígitos binários:

```
00 - 0
01 - 1
10 - 2
11 - 3
```

Lendo a porta A da PPI (com a função INP do BASIC) podemos descobrir em que slot está localizada cada página quando estamos trabalhando em BASIC. Isto é extremamente importante pois o padrão MSX não obriga o fabricante a localizar a RAM em algum slot específico. Não só fabricantes diferentes usam slot diferentes, como o mesmo fabricante pode optar por localizações distintas em versões diferentes do mesmo micro.

Existem certos programas que exigem o conhecimento do SLOT em que está cada página (veja dica 47, por exemplo).

Digite o programa da figura.36.2 e você obterá a distribuição das páginas do seu MSX quando estiver rodando em BASIC (lembre-se que, neste caso, a páginas 0 e 1 contêm o BIOS e o interpretador BASIC).

Figura.36.2

```
100 SCREEN 0:X=INP(&HA8)
110 PRINT "CONFIGURACAO DO SEU MSX"
120 FOR I=0 TO 3
130 P(I)=(X AND 3*(4^I))\(4^I)
140 P$(I)=STR$(P(I))
150 IF PEEK(&HFCC1+P(I))=0 THEN Y$(P(I))=
"   PRIMARIO":GOTO 190
160 IF PEEK(&HFCC1+P(I))=128 THEN A=PEEK(
&HFCC5+P(I))
170 Z(I)=(A AND 3*(4^I))\(4^I)
180 Y$(P(I))="."+RIGHT$(STR$(Z(I)),1)+" E
XPANDIDO"
190 PRINT "PAGINA";I;"-> SLOT ";P$(I)+Y$(
P(I))
200 NEXT I
```

Anote esta configuauração pois ela pode se tornar muito útil.

A linha 100 lê a Porta A da PPI(&HAB) e o laço 120-200 determina o númerb do slot de cada página.

Existem 4 variáveis (EXPTBL) nos endereços:

```
&HFCC1  (slot 0)
&HFCC2  (slot 1)
&HFCC3  (slot 2)
&HFCC4  (slot 3)
```

que normalmente contêm o valor 0 quando o slot

correspondente é primário (veja linha 150). Se o slot for expandido ,a variável correspondente contém o valor 128 (veja linha 160).

Neste caso, devemos consultar outras 4 variáveis (SLTTBL) :

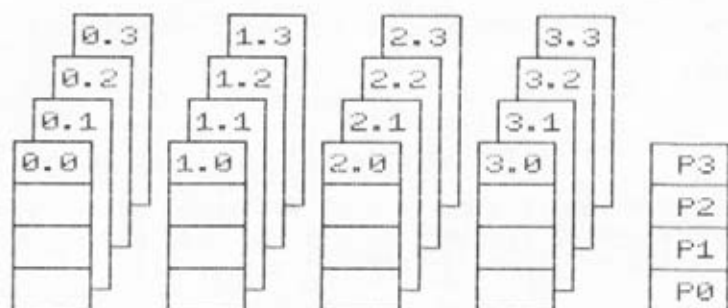
&HFCC5 (slot0)
&HFCC6 (slot1)
&HFCC7 (slot2)
&HFCC8 (slot3)

que duplicam o conteúdo dos registros de cada um dos possiveis 4 slots secundários, segundo uma codificação análoga à da porta A da PPI.

Cabe aqui lembrar que existem alguns softwares que rodam num determinado micro, mas não em outro com paginação diferente.

Lembre-se: em hipótese alguma podemos ou devemos responsabilizar o fabricante do micro. A norma MSX não exige nenhuma paginação particular.

O incompetente (ou imprevidente) é o programador do software(ou "adptador-pirata" em muitos casos) que não colocou uma rotina de identificação de paginação análoga a essa que acabamos de ver.

O software competente deve saber se localizar no micro em que está trabalhando: é isso que exige a norma MSX !

OS 16 SUB-SLOTS POSSÍVEIS DO MSX

P'RA QUE HEXADECIMAL?

37

Você deve ter notado que os programas em linguagem de Máquina que geram uma listagem BASIC cheia de linhas DATA, usam códigos hexadecimais. O mesmo ocorre com alguns endereços de RAM.

Se você está se iniciando em infomática, deve se perguntar, porque essa mania de usar notação hexadecimal?

Como se sabe, para usar um sistema de base N, deve usar N digitos diferentes.

No sistema binário, por exemplo, a base é 2 e os dígitos usados são 0 e 1.

No decimal, que estamos acostumados a usar desde crianças, a base é 10 e os 10 dígitos usados são 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9.

No hexadecimal (base 16) precisamos de 1 dígitos. Escolhem-se o 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B,C D e F .

Sim, você perguntará, mas qual é a vantagem?

Bem, como você sabe, a unidade de processamento do micro MSX é o byte, composto de 8 bits binários. Dispondo de X casas num sistema de base N você pode escrever N^x números diferentes.

Por exemplo com 3 casas, no sistema decimal, podemos escrever 10^3 números diferentes, que vão desde 000 até 999.

Nas placas dos automóveis, temos um prefixo de 2 letras (a escolher entre 26) . Podemos então formar 26^2=676 combinações diferentes de prefixos.

No byte podemos formar 2^8=256 números diferentes, que vão do 00000000 (=0) até o 11111111 (=255).

Usando a notação hexadecimal, podemos escrevr qualquer número contido num byte com apenas 2 dígitos (16^2= 256) .

Digite o programa da figura 37.1 para se familiarizar com as 3 notações:

Figura. 37.1

```
100 SCREEN0:KEY ON
110 KEY1,"DECIMAL"
120 KEY2,"BINARIO"
130 KEY3,"        "
```

```
140 KEY4,"HEXADEC"
150 KEY5,"        "
160 FOR I=0 TO 255
170 X$=MID$(STR$(I),2)
180 D$=RIGHT$("000"+X$,3)
190 B$=RIGHT$("00000000"+BIN$(I),8)
200 H$=RIGHT$("00"+HEX$(I),2)
210 PRINT D$;TAB(8);B$;TAB(24);H$
220 N=I\3
230 PLAY"L8N"+STR$(N)
240 IF PLAY(1) THEN 240
250 NEXT I
```

Na primeira coluna temos o possível conteúdo de um byte (de 0 a 255) em notação decimal, na segunda em binário e na terceira em hexadecimal.

Nas linhas até 150 usamos as teclas de função para titular as colunas (afinal o rodapé da tela é única coisa fixa).

Nas linhas 180, usando o RIGHT$, preenchemos com "zeros a esquerda" o número quando necessário e nas linhas de 220 a 240 fazemos o PSG emitir uma nota, tanto mais aguda quanto mais elevado for o número.

Cabe uma observação especial para a linha 170: quando usamos a função STR$ para transformar um número numa string de algarismos decimais, o primeiro caracter dessa string é sempre um espaço vazio .

Para eliminá-lo, usamos a função MID$ mandando formar numa string igual à obtida com STR$, a partir do segundo caracter (eliminando, portanto, o espaço vazio) de maneira a emendar os dígitos decimais com os "zeros á esquerda"da instrução seguinte .

Como você deve ter notado, analisando a tabela, o primeiro dígito hexadecimal determina os primeiros 4 bits do byte (1º nibble) e o segundo os últimos 4 (2º nibble) .

Assim sendo, se:

```
A= 1010
```

e

```
C= 1100
```

então

AC= 10101100

ou

CA= 11001010

Vamos então usar esta particularidade para aprender mais um truque.

Digite o programa da figura 37.2 para ver quais são os bits "acesos" e "apagados" de cada nibble .

Figura. 37.2

```
100 SCREEN 1
110 FOR I=224*8 TO 224*8+15
120  READ A$
130  B=VAL("&H"+A$)
140  VPOKE I,B
150 NEXT I
160 DATA FE,82,82,82,82,82,FE,00
170 DATA FE,FE,FE,FE,FE,FE,FE,00
180 FOR D=0 TO 15
190  H$=HEX$(D)
200  B$=RIGHT$("0000"+BIN$(D),4)
210  FOR K=1 TO 4
220   C=224+VAL(MID$(B$,K,1))
230   PRINT CHR$(C);
240  NEXT K
250  PRINT" = ";H$
260 NEXT D
```

A linhas 110 a 150 redefinem os caracteres de "alfa" (224) e "beta" da Screen 1 como sendo " quadrado vazio" (apagado) e "quadrado cheio" (aceso).

A forma dos caracteres está armazenado nas linhas DATA(160 e 170) de uma maneira que vamos entender daqui a pouco.

As linhas de 180 a 260 transformam os números de 0 a 15 em hexadecimal (H$) e 4 bits (B$) . A linha 220 associa ao bit 0, o caracter 224 (quadrado vazio) e ao bit 1, o carater 225 (quadrado cheio) . Você deverá obter a tabela da figura 37.3

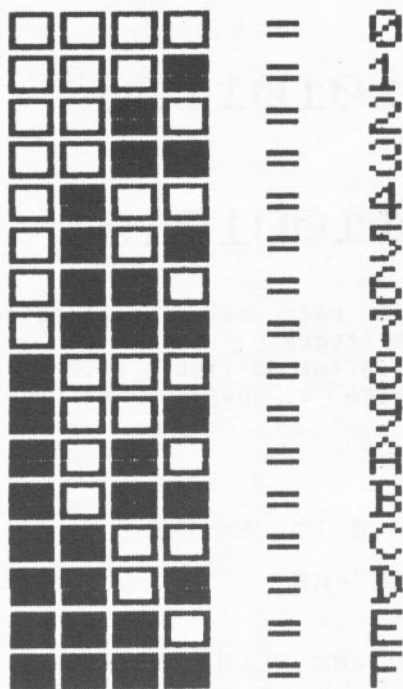
Figura. 37.3

Digamos, agora, que você queira definir um sprite com a forma dada pela figura 37.4

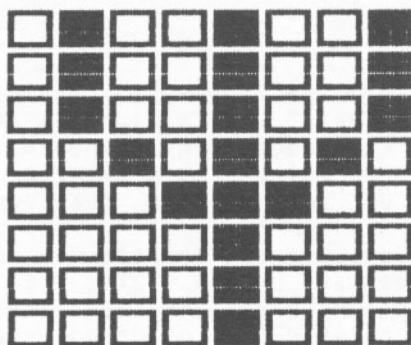
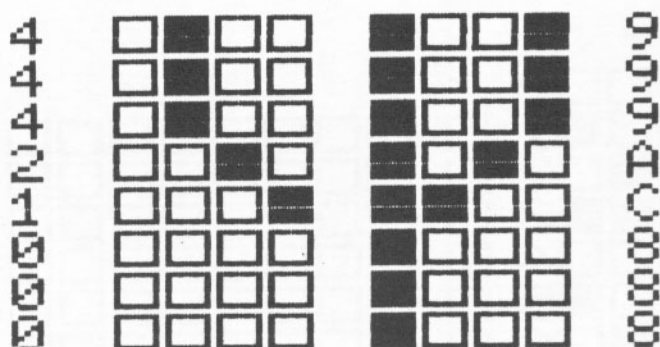
Figura.37.4

Inicialmente você deve dividí-lo ao meio e associar cada nibble ao seu código hexadecimal, consultando a figura 37.3

Figura. 37.5

Pronto! Associando o código da esquerda com o da direita, você tem a sequência de 8 bytes em hexadecimal que definem seu sprite:

```
49,49,49,2A,1C,08,08,08
```

Para definir este sprite, você poderia então usar o programinha de figura 37.6

Figura.37.6

```
100 REM ** DEFININDO SPRITE **
110 SCREEN 2,1
120 FOR I=1 TO 8
130  READ A$
140  S$=S$+CHR$(VAL("&H"+A$))
150 NEXT I
160 SPRITE$(0)=S$
170 DATA 49,49,49,2A,1C,08,08,08
200 REM ** POSICIONANDO SPRITE **
210 FOR ALFA =0 TO 50 STEP .1
220  X=100+ALFA*SIN(ALFA)
230  Y=100+ALFA*COS(ALFA)
240  PUT SPRITE 0,(X,Y),12,0
250  PSET(X,Y)
260 NEXT ALFA
```

```
300 REM ** CONGELA IMAGEM **
310 GOTO 310
```

Usando quadriculado (para desenhar sua matriz 8x8 da figura 37.7 você pode definir rapidamente qualquer sprite ou caracter.

Figura. 37.7

Se quiser, tire algumas xerox desta figura para usar mais vezes. Esta cópia é autorizada (veja Nota do Editor!) .

Uma última dica em relação à notação hexadecimal: como o endereçamento do Z80 (microprocessador do MSX) usa 2 bytes (16 bits), as regiões de memória vão de &H0000 A &FFFF . Ou seja, com dígitos você pode localizar qualquer endereço nos 64 Kbytes do MSX .

Algumas vezes, porém, você deve ter tentado descobrir quanto vale, em notação decimal, um endereço em hexa, e obtido valores negativos!

Isso se deve ao fato de que, os dois bytes da

numeração hexadecimal são usados para definir números inteiros com sinal. Sua correspondência, portanto não vai de 0 a 64K mas de -32K a +32K.

Digamos que você comande:

```
PRINT &HF34A
```

você deve obter

```
    -3256
```

que, em termos de endereçamento não significa nada!

O correto, para obter o endereço em decimal, é digitar :

```
PRINT &HF3*256+&H4A
```

obtendo:

```
   62282
```

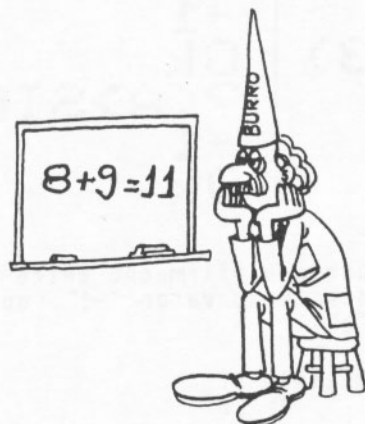
Se quiser saber em que página cai o endereço (veja dica 36).

Basta agora digitar

```
        PRINT INT (62282/16384)
```

No exemplo dado você obterá:

```
          3
```

ou seja, você está na página mais alta da memória.

38 USO DO PARÊNTESIS LóGICO

Um recurso muito poderoso mas pouco usado do BASIC MSX é o do parêntesis lógico.

Para entender seu funcionamento, ligue o micro e digite, no modo comando direto

A= 3 (e RETURN)

Neste momento, na memória do micro está armazenando o valor 3 na variável A. Tanto é que, se você comandar:

PRINT (A=3) (e RETURN)

o valor obtido agora é -1 !

Experimente digitar a seqüência de comandos diretos da figura 38.1 e tente descobrir a regra que está por trás disso:

Figura.38.1

```
A=3          ?(A=2)
Ok            0
?A           Ok
 3           ?(A>1)
Ok           -1
?(A=3)       Ok
-1           ?(A>SIN(1))
Ok           -1
             Ok
```

Percebeu? Quando a afirmação entre parêntesis é VERDADEIRA, ela devolve o valor "-1", quando é FALSA, devolve o valor "0":

(afirmação VERDADEIRA)= -1
(afirmaçao FALSA)= 0

Nem sempre a afirmação precisa vir entre parêntesis. Por exemplo, a função PLAY (veja dica N. 19) assume o valor -1(verdadeiro) enquanto o canal 1 está tocando e 0 (FALSO) quando ele está em silêncio .

Digite o programa da figura 38.2 para entender melhor o parêntesis lógico.

Figura. 38.2

```
100 SCREEN 0:KEY OFF
110 PI=4*ATN(1)
120 PRINT" PI          =";PI
130 PRINT"(PI>3)       =";(PI>3)
140 PRINT"(PI<3)       =";(PI<3)
150 PRINT"(INT(PI)=3)=";(INT(PI)=3)
160 PLAY"V15C16","V15A8","V15B4"
170 FOR I=1 TO 16
180 PRINT"PLAY(1)="PLAY(1);
190 PRINT"PLAY(2)="PLAY(2);
200 PRINT"PLAY(3)="PLAY(3)
210 NEXT I
```

Note que o Dó emitido no canal 1 é curto (C16) e obtém apenas 4 VERDADEIROS. O La do canal 2 (A8) tem o dobro de duração e obtém 8 VERDADEIROS enquanto que o Si do canal 3 (B4) tem o quádruplo da duração do DO e, consequentemente ganha 16 VERDADEIROS.

Vamos agora a uma aplicação mais prática dos parêntesis lógico.

Como você sabe, a função STICK do BASIC MSX lê as teclas de setas (ou os joysticks, assumindo os seguintes valores, em função da(s) tecla(s) pressionada(s) :

Figura. 38.3

```
8  1  2
 \ | /
7--0--3
 / | \
6  5  4
```

```
STICK(0)=TECLADO
STICK(1)=JOYSTICK A
STICK(2)=JOYSTICK B
```

Digite o programa da figura 38.4 e fique apertando uma tecla de setas (ou uma vertical E uma horizontal) para se familiarizar com a função STICK.

Figura. 38.4

```
100 KEY1,"teclar "
110 KEY2,"setas->"
120 KEY3,"  STICK"
130 KEY4,"  (0) ="
140 SCREEN 0:KEY ON
150 A=STICK (0)
160 A$=STR$(A)
170 KEY5,A$
180 GOTO 150
```

Agora que você entendeu o uso dos parêntesis lógicos e da função STICK, digite o programa listado na figura 38.5 e rode-o.

Para desenhar, aperte F1 (aparecerá o cursor #) e use as setas.

Para mover sem desenhar, aperte F2 (o cursor mudará para *).

Figura. 38.5

```
100 COLOR 15,1,1:X=100:Y=100
110 BASE(7)=BASE(9):SCREEN 1
120 SCREEN 2:OPEN "GRP:" AS #1
130 PRESET(0,0)
140 PRINT #1,"USO DO () LóGICO"
150 PRINT #1,"F1(#): MOVE DESENHANDO"
160 PRINT #1,"F2(*): MOVE SEM DESENHAR"
170 KEY(1) ON:KEY(2) ON
180 ON KEY GOSUB 500,600
190 A=STICK (0)
200 X=X-(A>1 AND A<5)+(A>5 AND A<9)
210 Y=Y-(A>3 AND A<7)+(A=8)+(A=1)+(A=2)
220 PUT SPRITE 0,(X-2,Y-4),CS,K
230 PSET(X,Y),CP
240 GOTO 190
```

```
500 REM F1 # MOVE DESENHANDO
510 CS=4:CP=15:K=ASC("#")
520 RETURN
600 REM F2 * MOVE SEM DESENHAR
610 CS=8:CP=1:K=ASC("*")
620 RETURN
```

Vamos à análise: a linha 100 define a cor de frente como BRANCO e a de fundo como PRETO (instrucão muito útil para quem tem um HOTBIT e usa um monitor monocromático) e define as coordenadas iniciais do desenho. A linha 110 cria uma tabela de sprites com a forma dos caracteres (veja dicas 4 e 5 para maiores detalhes desta "mãnha") .

As linhas de 130 a 160 escrevem mensagens na tela gráfica e a 170 ativa a interrupção para quando se teclar F1 ou F2. A 180 remete às subrotinas 500 e 600(move DESENHANDO e MOVE SEM DESENHAR) que simplesmente mudam a cor do sprite (CS), do ponto (CP) e a forma do sprite (# ou *) . Se você quiser, pode mudar a forma do "sprite-cursor" alterando os símbolos nas linhas 510 e 610.

A linha 190 lê as teclas de setas (se quiser mudar para usar o JOYSTICK, mude o STICK(0) para; STICK(1) ou STICK (2) para A e B respectivamente.

As linhas 200 e 210 atualizam os valores das coordenadas usando parêntesis lógicos. Note que, se por exemplo, você apertar a seta para direita (veja na figura 38.3 que ela tem o valor 3), a afirmação

```
(A>1 AND A<5)
```

é verdadeira e portanto vale-1. Em contrapartida, a afirmação:

```
(A>J AND A<9)
```

é falsa e vale 0.

Consequentemente, a atribuição

```
X=X-(A>1 AND A<5)+(A>5 AND A<9)
```

valerá:

```
X=100-(-1)+(0)=101
```

deslocando as coordenados de um ponto para direita.

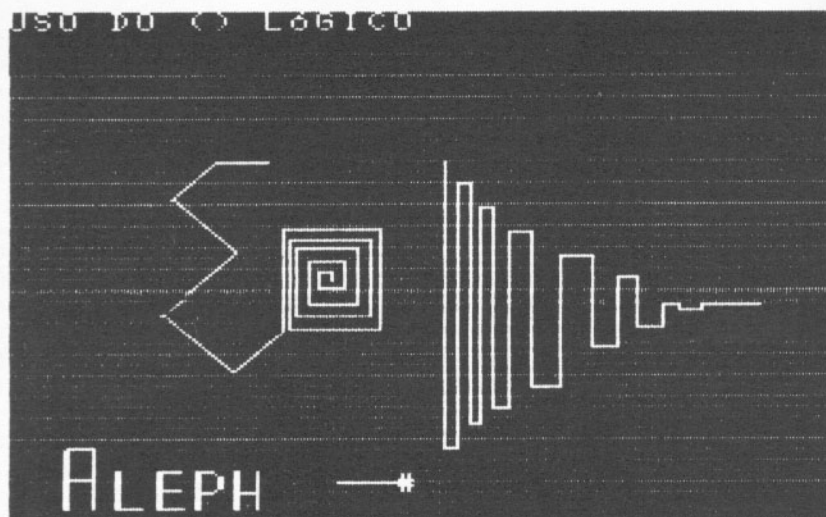
Só como exercício e consultando a figura 38.3, veja o que acontece com o X e o Y se você apertar

simultaneamente a tecla para cima e para esquerda!

Com este programinha você poderá gerar desenhos como o da figura 38.6

Figura.38.6

IMPRIMINDO TELAS DO MSX-LOGO

39

O cartucho de MSX-LOGO ou HOT-LOGO apresenta uma das mais bem sucedidas versões dessa linguagem.

Desenvolvida inicialmente para crianças, com um enfoque totalmente gráfico, ela permite a elaboração de figuras altamente complexas com extrema rapidez.

Novas versões de linguagens trazem, agora, uma versão meio simplificada de LOGO (o comando DRAW do BASIC é um claro exemplo disso!).

Porém, quem se rendeu aos "encantos" da tartaruguinha, não a troca por nada desse mundo para realizar seus desenhos.

O MSX-LOGO não permite que se possa imprimir as telas geradas, o que é meio frustrante para seu aficcionados, sejam eles crianças ou "marmanjos". Ter em mãos algo desenhado é muito mais gratificante do que apenas olhar algo na tela do computador.

O MSX-LOGO permite que se grave em disco as telas geradas. Para isso é usado o comando:

```
gravedes "nome do arquivo
```

Os "arquivos-tela" do LOGO possuem 13184 bytes e se você tentar carrega-lo com o comando BLOAD,S não conseguirá, pois o formato desse arquivo não é binário.

Sabemos que um "arquivo-tela" do LOGO possui, no mínimo as tabelas dos pontos (tabela de formação dos caracteres) e a do código das cores.

O programa em BASIC da figura 39.1 abre um arquivo randômico e transfere os primeiros 6144 bytes para a tabela de formação dos caracteres (do endereço 0 ao 6143 da VRAM) e os bytes restantes são transferidos para a tabela de cores (do endereço 8192 ao 14335 da VRAM).

O resultado é a transferência do desenho "padrão LOGO" para o padrão BASIC. A partir daí, você pode tirar cópias gráficas em sua impressora com o programa que mais lhe convier (consulte a dica 11 ou o "CEM DICAS PARA MSX").

Figura 39.1

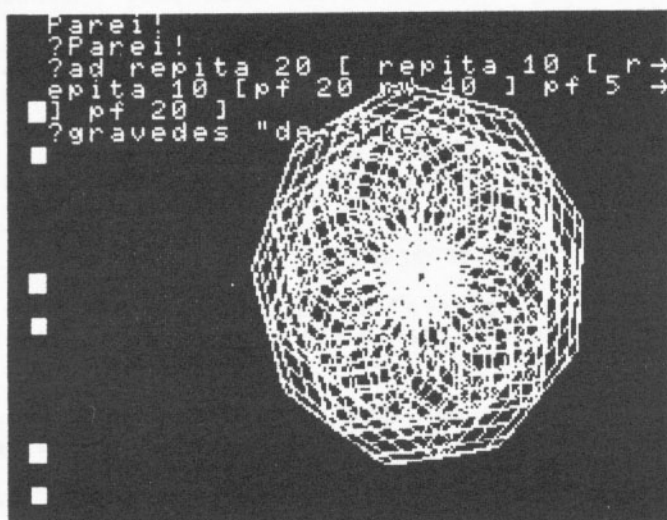
```
1000 DEFINT A-Z
1010 SCREEN 2
```

```
1020 OPEN "deslogo1" AS #1 LEN=1
1030 FIELD #1,1 AS A$
1040 D=-1
1050 FOR F=1 TO LOF(1)
1060 GET #1,F:VPOKE D+F, ASC(A$)
1070 IF F=6143 THEN D=8191-6143
1080 NEXT F:CLOSE #1
1090 BEEP :GOTO 1090
```

Veja na figura 39.2 um exemplo de tela gerada no LOGO.

Figura 39.2

CASANDO TELAS GRÁFICAS

40

Existem diversos editores gráficos para a linha MSX: EDDY2, GRAPHIC MASTER, GRAPHIC ARTIST e outros. Esses três editores (que são os mais importantes) possuem recursos bastante característicos, e muitas vezes é interessante usa-los alternadamente em fases distintas da edição da imagem.

As imagens geradas em disco pelo EDDY2 (com extensão ".SCR") podem ser lidas sem problemas pelo Graphic Master, e vice-versa. Também pode-se carregar as imagens ".SCR" através de um programa BASIC com o comando:

```
BLOAD "nome",S
```

e a imagem resultante na tela resulta perfeita.

O problema começa quando temos uma imagem gravada no disco com "BSAVE,S". Ao carregarmos a imagem no Eddy2 ou no Graphic Master, ela se encaixa no plano de fundo de maneira deficiente, ficando descentralizada e entrecortada. Isso se deve a um "offset" de 25 bytes que esses editores inserem no começo das telas, por motivos que escapam à nossa compreensão.

Para resolver este problema temos um programa que converte telas do padrão "normal" para o padrão "Eddy2" (que também vale para o Graphic Master), e do padrão "Eddy2" para o "normal".

A conversão do padrão "Eddy2" para o "normal" não é necessária para o comando BLOAD, mas o é para a tela poder ser lida pelo Graphic Artist. Neste editor, somente são reconhecidas as telas cuja extensão de nome seja ".GRP" (ufa!...).

Segue o programa na figura 40.1

Figura 40.1

```
10 REM CONVERSOR DE TELA P/ EDDY2
20 REM (C) 1987 BY THE PILOT
30 REM
50 SCREEN 0:INPUT "Nome do arquivo ";N$
60 PRINT:PRINT "1-> Normal->Eddy2"
70 PRINT "2-> Eddy2->Normal"
80 I$=INPUT$(1)
90 IF I$="1" THEN 120
```

```
100 IF I$="2" THEN 270
110 GOTO 80
120 BLOAD N$,&H9019
130 FOR F=&H9000 TO &H9018
140 POKE F,0
150 NEXT F
160 BSAVE N$,&H9000,&HC817
170 OPEN N$ AS #1 LEN=1
180 FIELD #1, 1 AS A$
190 LSET A$=CHR$(&HFE): PUT #1,1
200 LSET A$=CHR$(&HE7): PUT #1,2
210 LSET A$=CHR$(&HFF): PUT #1,3
220 LSET A$=CHR$(&HFF): PUT #1,4
230 LSET A$=CHR$(&H37): PUT #1,5
240 LSET A$=CHR$(&HE7): PUT #1,6
250 LSET A$=CHR$(&HFF): PUT #1,7
260 END
270 BLOAD N$,&H9000
280 BSAVE N$,&H9000,&HC7FE
290 OPEN N$ AS 1 LEN=1
300 FIELD 1,1 AS A$
310 LSET A$=CHR$(&HFE):PUT 1,1
320 LSET A$=CHR$(0):PUT 1,2:PUT 1,3
:PUT 1,6:PUT 1,7
330 LSET A$=CHR$(&H17):PUT 1,4
340 LSET A$=CHR$(&H38):PUT 1,5
350 END
```

Ele começa pedindo o nome da tela, que deve ser digitado inclusive com a extensão. Em seguida ele pergunta o sentido em que deve ser feita a conversão (Normal->Eddy2 ou Eddy2->Normal). Muito cuidado para não errar a opção. A tentativa de converter uma tela para o formato que ela já possui pode causar estragos nela.

Após a escolha, o programa carrega a tela na memória e a salva novamente no disco, deslocada de 25 bytes para frente ou para trás, de acordo com a opção dada.

A última etapa consiste em mudar os seis bytes de endereços do BSAVE (2º ao 7º do arquivo) para os valores corretos. O programa faz isso com os comandos OPEN e PUT (comandos para arquivo randômico).

RECUPERADOR DE ARQUIVOS.

41

Este programa recupera os arquivos deletados de um disco.

Ao deletar-se um programa/arquivo é feita apenas uma marcação de seu apagamento, e não o desaparecimento físico das informações nele contidas.

Esta marcação é a seguinte:

Troca-se o primeiro byte (Caracter) do nome do arquivo pelo caracter " à" (CHR$(229)).

Para recuperá-lo basta trocar o primeiro carater por uma letra.

Comandos.

<BARRA DE ESPAÇOS>- Seleciona a alteração de letra ou arquivo.

SETAS - Para a esquerda e direita mudam o arquivo a ser selecionado.
As setas "para cima" e "para baixo" mudam a letra inicial de um arquivo deletado.

[F1] - Grava todos os arquivos recuperados.

Funcionamento do programa:

O programa define as teclas de função com a mensagem ### TECLE ENTER ###.

O diretório do disco está entre os setores 5 e 11, sendo de até 16 arquivos por setor, totalizando assim, um máximo de 112 arquivos.

A instrução DSKI$ (DRIVE,SETOR) passa o setor do drive escolhido a partir do endereço indicado por uma variável de memória que está nos endereços &HF351 e &HF352.

O DSKO$ (DRIVE,SETOR) faz o inverso, ou seja, grava em disco o que está na memória.

O programa vasculha setor a setor a procura de arquivos deletados. Ao achar guarda o nome do programa e seu endereço em duas matrizes (N$ e N). Caso não encontre nenhum, apresenta uma mensagem de que não há arquivos apagados no disco.

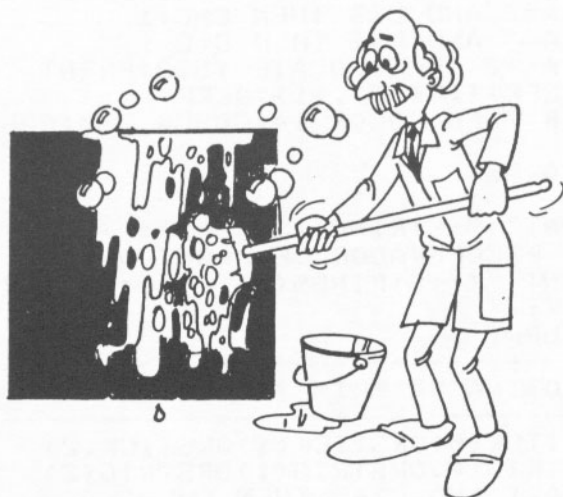
Depois de selecionar quais arquivos serão recuperados e teclar [F1] os programas recuperados terão seus nomes regravados no disco.

```
10 '    +-------------------------------+
20 '    !   Aldo Barduco Junior  - 88 !
30 '    +-------------------------------+
40 '    !   Recuperador  de  Arquivos   !
50 '    +-------------------------------+
60 CLEAR 2000:WIDTH 38
70 KEY1,SPACE$(6):KEY6,SPACE$(6)
80 KEY2,"    ###":KEY7,"    ###"
90 KEY3," TECLE":KEY8," TECLE"
100 KEY4,"ENTER ":KEY9,"ENTER "
110 KEY5,"###   ":KEY10,"###    "
120 DIM N$(112),N(112):KEY OFF
130 ON KEY GOSUB 800
140 COLOR 15,1,1:GOSUB 620
150 '-----------------------------------
160 ' TRAVA CAPS LOCK
170 '-----------------------------------
180 POKE &HFCAB,30
190 I$=INKEY$
200 IF I$<"A" OR I$>"D" THEN 190
210 PRINTI$,
220 PRINT SPACE$(9);"LENDO DIRETORIO"
230 '-----------------------------------
240 ' ROTINA DE LEITURA
250 '-----------------------------------
260 FOR H=5 TO 11
270 LOCATE 0,5:PRINT"LENDO SETOR:";USING
 "##";H
280 A$=DSKI$(ASC(I$)-64,H)
290 D=PEEK(&HF351)+256*PEEK(&HF352)
300 FOR G=0 TO 511 STEP 32
310 FOR F=0 TO 10
320 A$=A$+CHR$(PEEK(F+G+D))
330 NEXT F
340 IF LEFT$(A$,1)<>CHR$(229) THEN GOTO
390
350 S=S+1
360 N(S)=F+G+D-11
370 N$(S)=A$+CHR$(H):K$=LEFT$(N$(S),8)+"
."+MID$(N$(S),9,3)
380 LOCATE 16,5:PRINT"ARQUIVOS DELETADOS
:";USING "###";S
390 A$=""
400 NEXT G,H
410 IF S<>0 THEN 460 ELSE PRINT:PRINTSPC
(6);"NAO HA ARQUIVOS DELETADOS":FOR F=1
TO 70:IF F\3=F/3 THEN BEEP:NEXT
```

```
420 KEY ON
430 K$=INKEY$:IF K$<>CHR$(13) THEN 430
440 KEY OFF
450 RUN
460 KEY ON
470 K$=INKEY$:IF K$<>CHR$(13) THEN 470
480 KEY OFF
490 GOSUB 620:PRINTI$
500 '------------------------------------
510 ' ROTINA DE SELECAO
520 '------------------------------------
530 KEY (1) ON
540 C=1:A=1:L=65:GOTO 590
550 A=STICK(0)ORSTICK(1)ORSTICK(2)
560 B=STRIG(0)ORSTRIG(1)ORSTRIG(2)
570 IF A=3 AND C<S THEN C=C+1
580 IF A=7 AND C>1 THEN C=C-1
590 IF A<>0 THEN LOCATE 15,3:PRINT "ARQU
IVO: ";LEFT$(N$(C),11):BEEP
600 IF B THEN COLOR 14:GOSUB 760:GOSUB 6
70
610 GOTO 550
620 CLS
630 PRINT"+";STRING$(36,"-");"+";:PRINT"
!        RECUPERADOR DE ARQUIVOS       !";
640 PRINT "+";STRING$(36,"-");"+";:PRINT
"DRIVE=";
650 RETURN
660 '------------------------------------
670 ' ROTINA DA PRIMEIRA LETRA
680 '------------------------------------
690 A=STICK(0)ORSTICK(1)ORSTICK(2)
700 B=STRIG(0)ORSTRIG(1)ORSTRIG(2)
710 IF A=1 AND L>65 THEN L=L-1
720 IF A=5 AND L<90 THEN L=L+1
730 IF A<>0 THEN LOCATE 24,3:PRINTCHR$(L
):BEEP
740 IF B THEN N$(C)=CHR$(L)+RIGHT$(N$(C)
,11):COLOR 15:GOSUB 760:RETURN
750 GOTO 690
760 '------------------------------------
770 ' ROTINA DE PAUSA
780 '------------------------------------
790 FOR F=1 TO 100:NEXT:RETURN
800 '------------------------------------
810 ' ROTINA DE GRAVACAO
820 '------------------------------------
830 FOR F=1 TO S
```

```
840 IF LEFT$(N$(F),1)<>CHR$(229) THEN GO
SUB 890
850 NEXT F
860 FILESI$+":":KEY ON
870 K$=INKEY$:IF K$<>CHR$(13) THEN 870
880 KEY OFF:RETURN 10
890 A$=DSKI$(ASC(I$)-64,ASC(RIGHT$(N$(F)
,1)))
900 POKE N(F),ASC(LEFT$(N$(F),1))
910 DSKO$ASC(I$)-64,ASC(RIGHT$(N$(F),1))
920 RETURN
```

ORDENADOR DE DIRETóRIO.

42

Este programa ordena os arquivos do diretório de forma crescente. Quando temos um disco abarrotado de programas, para facilitar a visualização é funcional ordená-lo, pois a área de localização do arquivo fica menor, facilitando, sua localização.

Funcionamento do programa.

O diretorio de um disco encontra-se entre os setores 5 e 11. Cada setor tem 512 bytes. Cada arquivo ocupa 32 bytes (11 para o nome e mais 21 um para outras informacões), portanto pode-se ter um máximo de 16 títulos por setor, totalizando assim, 112 títulos em um disquete.

A leitura se faz com o comando DSKI$, que passa um setor do disco para a memória e a gravação com DSKO$, que faz o inverso.

O programa gera uma tabela na memória com todo o diretório. Isto consegue-se alterando o valor da variável de memória que está nos endereços &HF351 e &Hf352, que indicam onde serão colocados os dados lidos do disco.

O programa está listado na figura 42.1

Figura 42.1

```
10  '     +--------------------------+
20  '     ! Aldo Barduco Junior -88!
30  '     +--------------------------+
40  '     ! Ordenador de Diretorio !
50  '     +--------------------------+
60  '
70 DEFSNG A-Z
80 CLEAR 5000,&HBFFF:WIDTH 38
90 DIM N$(112),N(112):KEY OFF
100 KEY 1,SPACE$(6):KEY 2,SPACE$(6)
110 KEY 2,"   ###":KEY 7,"   ###"
120 KEY 3,"TECLE ":KEY 8,"TECLE "
130 KEY 4,"ENTER ":KEY 9,"ENTER "
140 KEY 5,"###":KEY 10,"###"
150 COLOR 14,1,1
160 '------------------------------------
170 ' TRAVA CAPS LOCK
180 '------------------------------------
190 POKE &HFCAB,&H30
```

```
200 GOSUB 890
210 I$=INKEY$
220 IF I$<"A" OR I$>"B" THEN 210
230 PRINTI$,
240 PRINT SPACE$(9);"LENDO DIRETORIO"
250 '------------------------------------------
260 ' ROTINA DE LEITURA
270 '------------------------------------------
280 FOR H=5 TO 11
290 E=49152!+(H-5)*512:E2=INT(E/256):E1=
E-INT(E/256)*256
300 POKE &HF351,E1:POKE &HF352,E2
310 A$=DSKI$(ASC(I$)-64,H)
320 FOR G=0 TO 511 STEP 32
330 FOR F=0 TO 31
340 A$=A$+CHR$(PEEK(F+G+E))
350 NEXT
360 IF LEFT$(A$,1)=CHR$(0) THEN 440
370 IF A$>"a" OR A$<"A" THEN 420
380 S=S+1
390 N(S)=F+G+E-32
400 N$(S)=A$:K$=LEFT$(N$(S),8)+"."+MID$(N
$(S),9,3)
410 IF S\2=S/2 THEN PRINT ,K$;"  SETOR:"
;USING"##";H ELSE PRINT K$;
420 A$=""
430 NEXT G,H
440 FOR F=1 TO 70:IF F\5=F/5 THEN BEEP
450 NEXT F
460 COLOR 15
470 KEY ON
480 K$=INKEY$:IF K$<>CHR$(13) THEN 480
490 KEY OFF
500 GOSUB 890:PRINTI$
510 '------------------------------------------
520 ' ROTINA DE ORDENACAO
530 '------------------------------------------
540 LOCATE 18,3:PRINT"ORDENACAO: CRESCENTE"
550 PRINT:PRINT
560 PRINT SPC(2);"EXECUTANDO ORDENACAO DO
 DIRETORIO"
570 FOR G=1 TO S-1
580 FOR F=G+1 TO S
590 IF N$(G)<N$(F) THEN 620
600 SWAP N$(G),N$(F)
610 Y=1
620 NEXT F
630 IF Y=1 THEN Y=0:NEXT G
```

```
640 IF G=1 THEN PRINT:PRINT:PRINTSPC(5);"
DIRETORIO JA ESTA ORDENADO":GOTO 820
650 BEEP:BEEP:COLOR 7
660 FOR G=1 TO S
670 FOR F=0 TO 31
680 POKE N(G)+F,ASC(MID$(N$(G),F+1,1))
690 NEXT F,G
700 BEEP:BEEP:COLOR 5
710 '------------------------------------
720 ' ROTINA DE GRAVACAO
730 '------------------------------------
740 GOSUB 890:PRINT I$,
750 PRINT SPC(6);"GRAVANDO DIRETORIO":PRINT
760 FOR H=5 TO 11
770 E=49152!+(H-5)*512:E2=INT(E/256):E1=E
-INT(E/256)*256
780 POKE &HF351,E1:POKE &HF352,E2
790 DSKO$ ASC(I$)-64,H
800 NEXT H
810 FILES
820 KEY ON
830 K$=INKEY$:IF K$<>CHR$(13) THEN 830
840 KEY OFF
850 RUN
860 '------------------------------------
870 ' CABECALHO
880 '------------------------------------
890 CLS
900 PRINT"+";STRING$(36,"-");"+";:PRINT"!
      ORDENADOR DE DIRETORIO          !";
910 PRINT "+";STRING$(36,"-");"+";:PRINT"
DRIVE=";
920 RETURN
```

43

QUAL O NOME DO DOS?

Como sabemos, o MSX pode operar com drives através de dois sistemas: o DISK-BASIC e o MSX-DOS. Quando ligamos uma interface de disco num slot do MSX, uma EPROM nela gravada contém um acréscimo ao sistema operacional e interpretador residentes do micro, tornando seu BASIC mais completo: o micro passa a dispor do DISK BASIC.

Através do DISK BASIC você pode gerenciar arquivos seqüenciais e randômicos (veja dica 48) , ler e escrever em setores de determinados pontos do disco, etc .

No DISK BASIC, porém ,você passa a ter menos memória disponível do que no BASIC sem disco, pois cada drive conectado (físico e/ou lógico) exige um "buffer" , ou seja, um espaço de trabalho na RAM, roubando um pouco de memória de que está disponível para o usuário .

Se você está, por exemplo, usando o cartucho do programa HOTWORD com um drive conectado, os dois "buffers" que ele reserva(um para drive A, outro para B) se sobrepõe ao programa e você não consegue gravar seus textos.

O truque, neste caso, é ligar o micro com a tecla CONTROL pressionada. Este procedimento indica ao sistema de interface, que queremos usar um único drive, e o programa no cartucho passa a funcionar o contento.

Em contrapartida, o MSX-DOS é um sistema operacional que é carregado a partir do disco (ele entra assim que o micro é ligado e tenta ler o drive), que "abre" os 64 Kbytes de memória do MSX.

Se você tem um único drive e tentou copiar um lote de arquivos do drive lógico A para o drive lógico B , já deve ter sentido a imensa diferença em realizar este operação via DISK-BASIC ou via MSX-DOS .

No MSX-DOS, muitos arquivos são lidos e colocados na memória RAM. Só então é que o sistema solicita a troca do disco-fonte pelo disco-destino.

No DISK-BASIC porém, você passa a ter uma frequência enloquecedora.

Portanto uma dica importante é carregar o DOS do disco antes de fazer um grande volume de cópias.

Neste ponto surge um problema: digamos que você

tenha formatado seu disco numa interface e que tenha copiado um DOS de outro disco.

Como a LEI de Informática proíbe pagar "royalties" para a MICROSOFT (autora do MSX-DOS), cada fabricante brasileiro, deu ao seu DOS um nome especial.

Acontece que, ao formatarmos um disco, a rotina de formatação gera, em sua trilha 0 informações e um curto programa em Linguagem de Máquina (o BOOT) que, entre outras coisas, procura o DOS pelo nome que está gravado na trilha zero.

Para melhor entender isso, digite o programa da figura 43.1.

A linha 110 copia a trilha 0 do disco e a coloca na RAM numa região cujo endereço é apontado pelo varáveis do sistema que estão em &HF351 (LSB) e &HF352 (MSB) .

A linha 120 calcula o valor deste endereço e o laço 130-250 imprime na tela os caracteres correspondentes ao conteúdo desta região de RAM (cópia fiel da trilha 0 do seu disco).

O restante das linhas serve para formatar esta tabela de maneira a podermos descobrir a posição de cada caracter por notação hexadecimal (veja dica 37).

Figura 43.1

```
100 SCREEN 1:WIDTH 32:KEYOFF
110 A$=DSKI$(0,0)
120 EN=PEEK(&HF351)+256*PEEK(&HF352)
125 PRINT"  ";
130 FOR I=0 TO 15
140 PRINT HEX$(I);
150 NEXT I
160 PRINT:PRINT
170 FOR F=0 TO 255 STEP 16
180 PRINT HEX$(F/16);" ";
190 FOR K=0 TO 15
200 X=EN+F+K
210 C=PEEK(X)
220 IF C<32 THEN PRINT CHR$(1)+CHR$(64+C)
;ELSE PRINT CHR$(C);
230 NEXT K
240 PRINT
250 NEXT F
260 LOCATE 0,19
270 PRINT"SETOR ZERO"
```

Se você rodar este programa com um disco formatado numa interface japonesa, obterá a tabela da figura 43.2

Figura 43.2

Como você pode notar, o nome do DOS (MSXDOS SYS) começa em A0 e termina em AA.

Se você rodar o proprama com um disco formatado pela MICROSOL (primeiro fabricante de drives para MSX no BRASIL) você obterá a tabela da figura 43.3

Figura 43.3

Neste, o nome MSXDOS SYS começa em A8, mas existe outro nome: SOLXDOS SIS, que começa em CE. Este boot (que foi parcialmente refeito: você pode notar isso

comparando com a figura 43.2) inicialmente procura um DOS com o nome SOLXDOS.SIS. Não o encontrando, opta pelo MSXDOS.SYS.

O mesmo sistema foi adotado pelas interfaces da TROPIC (DISPRO.SYS) e de DDX (DDXDOS.SYS), mostrando que estes sistemas foram pirateados a partir de um original não japonês mas (pasmem!) cearense!

Veja as figuras 43.4 e 43.5

Figura 43.4

Figura43.5

Mas, de qualquer forma, eles também permitem carregar o MSXDOS.SYS padrão se não encontrarem um nome tupiniquim!

Pior que isso fez a SHARP: veja a figura 43.6 e compare com a tabela da interface japonesa!

Figura43.5

Além de alardear aos 4 ventos que seu sistema era original (com cópia do software registrado na S.E.I e tudo mais!) colocaram como única opção o nome HBDOS.SYS.

Isso quer dizer que um disco formatado numa interface SHARP padrão MSX não carrega o DOS padrão MSX!

Qual a solução para essa bagunça? Você dirá "eu estou com um disco formatado na interface X e um BABADOS de origem Y! Com faço X para carregar esse BABADOS ? "

Fácil. rode o programinha da figura 43.1 e veja qual o nome do DOS na tabela gerada na tela: digamos que seja XYZDOS.SYS enquanto que o DOS que você tem gravado, seja BABADOS.SUS.

Basta então comandar:

```
NAME "BABADOS.SUS" AS "XYZDOS.SYS"
```

Tire o disco do drive, desligue o micro, ligue de novo recoloque o disco,. Provavelmente o DOS vai entrar, com alguma orgulhosa mensagem do fabricante, se achando muito "original" !

ARQUIVOS BATCH DO MSX-DOS.

44

O MSX-DOS e seus clones (SOLXDOS, DDXDOS, HBDOS, etc) possuem um editor de linhas que é pouco conhecido.

Ele é tratado como um dispositivo dentro do DOS. Da mesma forma que temos o nome A: para o drive A, temos o nome CON para o CONsole. Ou seja, o teclado do micro.

A principal finalidade do dispositivo CON é a de criar arquivos BATCH.

Um arquivo BATCH (que quer dizer lote) é um seqüencia de comandos do MSX-DOS no formato ASCII. Para que um arquivo BATCH seja reconhecido pelo DOS, ele precisa, obrigatóriamente, possuir a extensão .BAT.

Um arquivo BATCH pode ser criado por qualquer editor de texto que armazene o texto apenas no formato ASCII. Porém, a maneira mais fácil e rápida de fazê-lo é através do comando COPY (do DOS) e do dispositivo CON.

Estando no MSXDOS, digite o seguinte comando:

```
COPY CON TESTE1.BAT
```

Ao pressionar a tecla RETURN, você verá o nome TESTE1.BAT e o sinal de "A>" característico do DOS desaparecerá. Você estará no dispositivo CON.

Experimente digitar os comandos da figura 44.1. Lembre-se de digitar RETURN após o final de cada linha.

O ^Z deve ser digitado pressionando-se a tecla CTRL e a tecla Z conjuntamente. Ele indica que é o final do arquivo.

Figura 44.1

```
MODE 40
REM ****> DISCO DE TRABALHO <****
PAUSE
DIR /W
^Z
```

Se você digitou corretamente, deve aparecer a mensagem indicando que 1 arquivo foi copiado.

A seguir digite:

```
TESTE1
```

Você verá que cada linha do arquivo TESTE1.BAT será executada como se tivesse sendo digitada.

Qualquer comando do MSXDOS pode ser usado. Até mesmo o nome de programas podem estar em um arquivo BATCH.

Se ao criar um arquivo BATCH você nomea-lo com:

```
AUTOEXEC.BAT
```

o lote de comandos nele contido será executado assim que for feito o "boot" do disco.

Experimente agora comandar:

```
COPY TESTE1.BAT CON
```

Você obterá na tela o arquivo TESTE1.BAT. Esse é um modo diferente de conseguir ler o conteúdo de um arquivo. O funcionamento é idêntico ao do comando TYPE do DOS.

OS ARQUIVOS ".COM" DO MSX-DOS

45

O padrão CP/M, difundido no mundo inteiro, obteve grande sucesso porque foi o primeiro sistema padronizado para microcomputadores.

Todas as rotinas de entrada e saída (vídeo, impressora, drive, etc) respeitam certos parâmetros de entrada ou saída. É por isso que tanto um TRS-80 como um MSX conseguem rodar os programas padrão CP/M.

O MSX-DOS é um programa compatível com o CP/M 2.2 e ainda mantém a grande vantagem de não alterar a formatação do disco (como faz o HB-MCP da SHARP).

Os programas padrão CP/M que possuem a extensão .COM, são obrigatóriamente programas em Linguagem de Máquina que são carregados e executados a partir do endereço 0100H da memória (lembre-se que em MSX-DOS temos disponíveis os 64 Kbytes de RAM).

Para trabalharmos em MSX-DOS precisamos que o disco no qual o BOOT foi realizado contenha o DOS (ou pelo menos um de seus "clones" nacionais).

O MSX-DOS sempre vem acompanhado de um arquivo chamado COMMAND.COM. É nesse arquivo que estão os commandos do DOS (TYPE, COPY, PAUSE, etc). Rode o programa da figura 45.1. Você verá um monte de caracteres estranhos, mas também, entre eles, o nome dos comandos disponíveis no DOS.

Figura 45.1

```
100 OPEN"A:COMMAND.COM" AS #1 LEN=1
110 FIELD #1,1 AS A$
120 SCREEN 0
130 N=N+1
140 GET#1,N
150 A= ASC(A$)
160 IF A<32 THEN PRINT CHR$(1)+CHR$(64+A);
 ELSE PRINT CHR$(A);
170 IF N=LOF(1) THEN END
180 GOTO 130
```

Se você possui apenas o SOLX-DOS, não se preocupe, altere a linha **100** para:

```
100 OPEN "SOLXDOS.SIS" AS #1 LEN=1
```

A Microsol, ao elaborar o seu "DOS" juntou, em um mesmo arquivo, o equivalente ao MSX-DOS e COMMAND.

Sempre que o MSX-DOS é chamado, ele prepara a memória com a padronização do CP/M e chama o programa COMMAND.COM em seguida.

Se você observar bem, o COMMAND possui a terminação .COM, o que quer dizer que ele é um programa em Linguagem de Máquina que é executado a partir do endereço 0100H. Ou seja, o formato dele é idêntico ao de um programa compilado em PASCAL, ou ao WORDSTAR, dBASE ou qualquer programa .COM que você possuir.

Se o DOS estiver instalado, podemos "pular" do BASIC para o DOS e vice-versa com os comandos:

BASIC **(do DOS para o BASIC)**

CALL SYSTEM **(do BASIC para o DOS)**

Porém, não podemos fazer com que a execução de algum programa do DOS seja automática quando comandamos um CALL SYSTEM.

Lembrando que:

O COMMAND e qualquer programa .COM possuem o mesmo formato.

O MSX-DOS é chamado no BOOT e no CALL SYSTEM.

O MSX-DOS chama o programa COMMAND.COM.

Se você quiser obrigar a execução de um programa .COM basta mudar o nome do COMMAND.COM para outro qualquer e renomear o programa que você quer executar para COMMAND.COM.

Instale o DOS no micro, chame o Interpretador BASIC comandando BASIC e digite o programa da figura 45.2

Figura 45.2

```
100 NAME"COMMAND.COM" AS "CMD.COM"
110 INPUT"Qual o nome do novo COMMAND.COM
";N$
120 NAME N$ AS "COMMAND.COM"
130 CALL SYSTEM
```

Esta dica é interessante para ser usada quando você possui um "COMMAND alternativo" (como o programa

MENU.COM do MSX-DOS-TOOLS), se você mesmo quer elaborar um DOS ou se você quer a execução automática de um programa e não quer usar um arquivo BATCH.

```
DIR         1C
COPY        TYPE        REN         DEL         V
DATE        TIME        RENAME      ERASE
PAUSE       REM         VERIFY      MODE
 BASIC      FORMAT      Insert disk wi
th batch file$ and strike any key when
 ready  $Strike a key when ready . . .
$Invalid drive specification$Program to
o big to fit in memory$ file$ bytes fre
e$Bad command or file name  $File not f
ound$Are you sure (Y/N)? $Rename error$
Invalid parameter$Insufficient disk spa
ce  $File creation error  $File cannot
be copied onto itself  $Content of dest
ination lost before copy  $ copied$Writ
e error  $Current date is $  Invalid da
te$  Enter new date: $Current time is $
  Invalid time$  Enter new time: $SunMo
nTueWedThuFriSat  <DIR> $COMBAT
```

46 DIRETÓRIO NA SCREEN 2

Este programa mostra o diretório na tela de alta resolução em 64 colunas. Se combinado ao programa de copia gráfica (livro: 100 dicas pág.127), poderá gerar etiquetas para a capa de disquetes.

É um "pouco" lento, pois foi desenvolvido em BASIC, mas permite a visualização em apenas uma tela de todos os arquivos de um disco.

Funcionamento do Programa:

A montagem dos caracteres (ver prog. 47.2) é feita a partir da tabela de caracteres "normais". . A nova tabela gerada é colocada apartir do endereço 48000 (&HBB80).

Como o SLOT onde encontra-se a RAM varia de micro para micro, temos que procurá-lo (veja dica 36).

Os caracteres são impressos na tela de alta resolução saltados de 4 em 4 pontos, já que os caracteres ocupam apenas os 3 primeiros bits de cada byte.

A leitura do diretório se faz setor a setor através dos comandos do BASIC, DSKI$ e DSKO$ (ver dicas 41 e 42), que fazem respectivamente, leitura e gravação.

Figura 46.1

```
10 '     +------------------------------+
20 '     ! Aldo Barduco Junior -88!
30 '     +------------------------------+
40 '     ! Diretorio na Screen 2  !
50 '     +------------------------------+
60 '
70 '----------------------------------------
80 ' MONTA TABELA DE CARACTERES 64
90 '----------------------------------------
100 SCREEN 1:PRINT"MONTANDO TABELA DE CA
RACTERES"
110 FOR F=256 TO 727
120 A=VPEEK(F)
130 B=A AND 24:C=A AND 192:D=A AND 32
140 IF B<>0 THEN B=32
150 IF C<>0 THEN C=128
160 IF D<>0 THEN D=64
170 POKE &HBB80+F, (B OR C OR D)
```

```
180 NEXT F
190 POKE &HBD00,64:POKE &HBD06,64:POKE &H
BD90,192:POKE &HBD96,192:POKE &HBDF2,224
:POKE &HBDF4,224
200 KEY OFF:SCREEN 0
210 DEFSNG A-Z:CLEAR 5000:WIDTH 38
220 DIM N$(112)
230 '------------------------------------
240 ' SLOT DA RAM
250 '------------------------------------
260 PPI=INP(&HA8)
270 A=(PPI AND 48)/16
280 '------------------------------------
290 ' FAZ 64 COLUNAS
300 '------------------------------------
310 POKE &HF920,&H80
320 POKE &HF921,&HBB
330 POKE &HF91F,A
340 '------------------------------------
350 ' TRAVA CAPS LOCK
360 '------------------------------------
370 POKE &HFCAB,&H30
380 GOSUB 860
390 I$=INKEY$:IF I$<"A" OR I$>"B" THEN 390
400 PRINTI$:BEEP
410 '------------------------------------
420 ' ROTINA DE LEITURA
430 '------------------------------------
440 D=PEEK(&HF351)+256*PEEK(&HF352)
450 FOR H=5 TO 11
460 LOCATE24,3:PRINT"LENDO SETOR:";USING"
##";H
470 A$=DSKI$(ASC(I$)-64,H)
480 FOR G=0 TO 511 STEP 32
490 FOR F=0 TO 31
500 A$=A$+CHR$(PEEK(F+G+D))
510 IF A$=CHR$(0) THEN 580
520 IF A$>"a" OR A$<"A" THEN 560
530 NEXT F
540 S=S+1
550 N$(S)=LEFT$(A$,8)+"."+MID$(A$,9,3)
560 A$=""
570 NEXT G,H
580 '------------------------------------
590 ' DIRETORIO NA SCREEN 2
600 '------------------------------------
610 SCREEN 2
620 OPEN"GRP:" AS #1
```

```
630 FOR F=1 TO S
640 A$=N$(F)
650 FOR I=1 TO LEN(A$)
660 PRESET ((I-1)*4+C*52,L*8):PRINT #1,MI
D$(A$,I,1)
670 NEXT I
680 C=C+1:IF C>4 THEN C=0
690 L=L-(F/5=F\5)
700 NEXT F
710 A$=STR$(S)+" ARQUIVOS "
720 LINE(155,176)-(LEN(A$)*4+156,187),,B
730 FOR I=1 TO LEN(A$)
740 PRESET ((I-1)*4+156,178):PRINT #1,MID
$(A$,I,1)
750 NEXT I
760 FOR F=1 TO 50:IF F/5=F\5 THEN BEEP
770 NEXT F
780 I$=INKEY$:IF I$<>CHR$(13) THEN 780
790 '-----------------------------------
800 ' FAZ 38 COLUNAS
810 '-----------------------------------
820 POKE &HF920,&HBF
830 POKE &HF921,&H1B
840 POKE &HF91F,0
850 SCREEN0:RUN210
860 '-----------------------------------
870 ' CABECALHO
880 '-----------------------------------
890 CLS
900 PRINT"+";STRING$(36,"-");"+";:PRINT"!
       DIRETORIO NA SCREEN II        !";
910 PRINT "+";STRING$(36,"-");"+";:PRINT"
DRIVE=";
920 RETURN
```

```
FAIXA   .58 FAIXA2  .TXT FAIXA1  .TXT N       .    FM1     .
FM2     .   FM3     .    FM4     .    FM5     .    FM6     .
FM7     .   FM8     .    FM9     .    FMA     .    FMB     .
XSME000 .BIN IMP     .BIN ALT     .ASM MSXTURBO.BAS MSXTURBO.BIN
MSXTEL  .COM PETALAS .    CONVEDDY.58 B       .    EDRAN7  .
DUVIDA2 .    HOOKCH  .58  REATBA  .58 DISCAD  .58  LABIR   .58
PAINT   .58  TITULO  .58  COPYBAS2.58 COPYBAS4.58  EDSPRITE.58
EDDRAN  .58  REC-ARB .58  REATBA  .BAS REATBA  .BIN FILTRO2 .BIN
VALPIER .58  ORDISK  .58  DELTA1  .   DELTA2  .    FIGBUF  .
DELTA3  .    DELTA4  .    LABIR   .BAS FAIXA50 .BAS FAIXA   .TXT
NUVEM1  .    NUVEM2  .    NUVEM3  .   TIT     .BAS PAINT   .BIN
FORIMP  .TXT LETER64 .BAS ITALIZA .BAS ITALIZA2.BAS ITALIZA3.BAS
GERTABS .BAS GERTABS2.BIN GERTABS .ASM BUFFER  .TXT A       .DRN
DIR-II  .58  ASC-ROLL.58  SCRSLT  .BAS SLOTPIER.2   TESTESL .
COPIA24 .TXT FIG62   .SCR STROBE  .BAS STROBE2 .BAS STROBE3 .BAS
IMPLIGAD.TXT EDEMERG .BAS GERAED  .BAS EDKK    .BAS EDCASEIR.TXT
MOUSE   .TXT LSTTEL  .BAS LSTTEL2 .BAS TABULAC .TXT TABVID  .BAS
DIRS2   .BAS MOUSE   .BAS KK      .KK  LABIR2  .BAS FFF     .DRN
ALDO    .SPR ALDO    .DRN OLIVER  .DRN ORDISK  .TXT YDOSPRIT.TXT
PAGMEM  .TXT CACA 1  .
                                   97 ARQUIVOS
```

LOCALIZANDO TABELAS DE CARACTERES. 47

Como você já viu na dica 23, podemos usa algorítmos simples para gerar rapidamente tabelas de caracteres alternativos.

Digite o programa da figura 47.2 mas, por enquanto, não o rode.

Na linha 300 precisamosinformar ao micro em que slot está a RAM. Você pode descobrir isso usando a dica 36.

Digamos que você esteja usando um Expert 1.1 fabricado em 1988. Rodando o programa da dica 36 você obteve o indicado na fig. 47.1

O byte **FXXXSSPP** que oferece na linha 240 deve ser corretamente preenchido para que o programa fique adequado ao seu micro. Como queremos gravar nossa tabela a partir do endereço &HBB80 (finalzinho da página 2 e eventualmente começo da 3) precisamos saber em que slot estão essas duas páginas.

Figura. 47.1

```
CONFIGURACAO DO SEU MSX
PAGINA 0 -> SLOT  0   PRIMARIO
PAGINA 1 -> SLOT  0   PRIMARIO
PAGINA 2 -> SLOT  2   PRIMARIO
PAGINA 3 -> SLOT  2   PRIMARIO
```

Como, no nosso exemplo, elas estão em slot primário, o bit F deve ser substituido por 0. Se os slots fossem secundários preencheríamos com 1.

Os bits XXX são irrelevantes e podem ser substituidos por 000.

O par PP (ou SS se o slot for secundário) devem conter 2 bits que dêem o valor do número do slot .

Lembre-se que, em binário:

&B00=0
&B01=1
&B00=2
&B11=3

No nosso exemplo, substituiremos PP por 10 (slot 2) e SS por 00.

Se você tivesse obtidos as páginas 2 e 3 no slot 3 secundário, substituiria o SS por 11 (slot 3) e o PP por 00.

No nosso exemplo, então, a linha ficará:

```
300 POKE&HF91F, &B00000010
```

Agora, após adequar a linha 240, pode rodar seu programa.

Figura.47.2

```
100 SCREEN 1:PRINT"TESTE":DEFINT A-Z
110 FOR F=ASC(" ")*8 TO ASC("Z")*8+8
120  A=VPEEK(F)
130  B=A AND &B00011000
140  C=A AND &B11000000
150  D=A AND &B00100000
160  IF B<>0 THEN B=&B00100000
170  IF C<>0 THEN C=&B10000000
180  IF D<>0 THEN D=&B01000000
190  POKE &HBB80+F, (B OR C OR D)
200 NEXT F
210 POKE &HBB80+ASC("0")*8,  &B01000000
220 POKE &HBB80+ASC("0")*8+6,&B01000000
230 POKE &HBB80+ASC("B")*8,  &B11000000
240 POKE &HBB80+ASC("B")*8+6,&B11000000
250 POKE &HBB80+ASC("N")*8+2,&B11100000
260 POKE &HBB80+ASC("N")*8+4,&B11100000
270 A$=INPUT$(1)
280 POKE &HF920,&H80
290 POKE &HF921,&HBB
300 POKE &HF91F,&BFXXXSSPP
310 SCREEN 2:OPEN"GRP:" AS #1
320 A$="EDITORA ALEPH - (011) 843-3202"
330 FOR K=1 TO LEN(A$)
340  PRESET (10+4*K,40)
350  PRINT #1,MID$(A$,K,1)
360 NEXT K
```

```
370 LINE(6,30)-(4*LEN(A$)+21,55),,B
380 GOTO 380
```

Enquanto ele está sendo executado, vamos analisá-lo.

A linha 100 chama a SCREEN 1 e portanto transfere a tabela de caracteres da ROM (a partir do endereço &H1BBF, slot 0) para o endereço 0 da VRAM.

O laço 120-200 lê os caracteres da VRAM, os faz passar pelo algorítmo "estreitador" da dica 26 e os transfere para uma tabela da RAM localizada a partir do endereço &HBB80.

Note que , para abreviar o exemplo, só transferimos desde o espaço vazio até o Z maiúsculo. Você pode alterar a linha 110 de maneira a transferir mais ou menos caracteres.

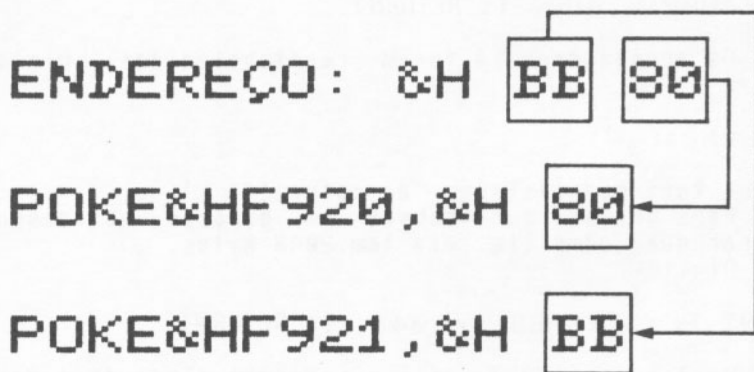
As linhas 210 a 260 dão uma "ajeitadinha" nos caracteres 0, B e N, pois o algoritmo é um pouco grosseiro e gera ambigüidades.

A linha 270 apresenta o cursor e aguarda a digitação de uma tecla, veja o que acontece e continue lendo esta análise.

As linhas 280 a 300 são críticas: elas mudam os apontadores do BASIC para a nova tabela da RAM.

Como você notou, o endereço da nova tabela em hexadecimal é constituido por 4 dígitos em hexadecimal (no nosso caso &HBB80).

Os últimos 2 devem ser "pokeados" no endereço &HF920 e os dois primeiros em &HF921 (fig 47.3)

Figura. 47.3

Obviamente, se você tivesse resolvido localizar sua tabela em, por exemplo, &HBA7F, as linhas 280 e 290 seriam:

```
280 POKE&HF920,&H7F
290 POKE&HF921,&HBA
```

A linha 300 já foi analisada.

As linhas de 310 a 370 escrevem uma mensagem na SCREEN 2 com a nova tabela de caracteres. Note que agora cabem 64 caracteres por linha pois o incremento no PRESET (linha 340) é de 4 em 4.

Dê agora um CONTROL + STOP (aparecerão abobrinhas pois as minúsculas não foram redefinidades) e comande um LIST.

Você terá uma listagem com as letras maiúsculas redefinidas. E agora?

Como volto ao normal? Fácil, basta lembrar que a tabela residente está no endereço &H1BBF do slot 0 (ROM).

Trave a CAPSLOCK (para só digitar maiúsculas) e comande:

POKE&HF920,&HBF(e RETURN)

leve o cursor novamente até esta linha e altere-a para:

POKE&HF921,&H1B (e RETURN)

leve novamente o cursor para essa linha e altere-a para:

POKE&HF91F,&H00 (e RETURN)

Os apontadores já foram reestabelecidos. Digite agora:

```
SCREEN0:LIST
```

e você terá sua listagem "ao natural"!

Para gravar sua tabela em binário no disco, lembrar que, completa, ela tem 2048 bytes.

Digite:

PRINT HEX$ (&HBB80+2047) (e RETURN)

Você obterá C3F7 que é o endereço final de tabela

completa na RAM.
Comande, então:

```
BSAVE "TABELA1. BIN",&HBB80,&HC3F7
```

e ela ficará gravada no disco.
Para chamá-la, basta comandar:

```
BLOAD "TABELA 1.BIN"
```

e ela se localizará na RAM.
Para ativá-la, seu programa deve ter as linhas

```
XXX POKE&HF920,&H80
YYY POKE&F921,&HBB
ZZZ POKE&HF91F,&BFXXXSSPP
```

e um comando SCREEN em seguida.
Para finalizar, vamos aprender mais um truque: digamos que você já tenha outra tabela na RAM (em &HBB80) e que você não queira perder ao carregar esta.
Sua TABELA 1.BIN pode ser então carregada 2Kbytes acima da já existente.
Basta usar o "SHIFT" do BLOAD e comandar:

```
BLOAD "TABELA 1.BIN" , 2048
```

e ele será caregada a partir do endereço &HC380 pois:

```
PRINT HEX$ (&HBB80 + 2048)
```
(e RETURN)

resulta em C380.
Obviamente, para ativar sua tabela assim deslocado, seu programa deve conter as linhas

```
XXX POKE&HF920,&H80
XXX POKE&HF921,&HC3
ZZZ POKE&HF91F,&BFXXXSSPP
```

Note que, com este processo, você pode alternar 2 ou mais tabelas de caracteres em seu micro !

48

Uma boa quantidade de usuários de MSX nos telefonam ou escrevem por terem dúvidas sobre a estrutura de arquivos sequenciais e randômicos.

Vamos então construir alguns curtos programas para entender melhor esta estrutura.

Inicialmente vamos abrir, no disco, um curto arquivo sequencial, para analisar sua estrutura.

Digite o programa de figura 48.1

Figura48.1

```
100 OPEN"A:LISTA.SEQ" FOR OUTPUT AS #1
120 FOR I= 1 TO 5
130 INPUT A$
140 PRINT #1,A$
150 NEXT I
160 CLOSE
```

Rode o programa e, ao ser solicitado pelo ponto de interrogação digite 5 nomes e telefones conforme a figura 48.2

Figura48.2

```
run
? ANA 240-8866
? SOLANGE 543-1147
? SUZY 64-7898
? JACQUELINE 290-9169
? FLAVIA 240-1131
Ok
█
```

A linha 100 abre um arquivo no drive para saída (FOR OUTPUT) . Você deve ver o LED do drive acender .

O laço 120-150 pede 5 nomes (INPUT A$) e os escreve no arquivo #1 que acabamos de abrir (PRINT#1,A$) .

A linha 160 fecha o arquivo (o LED acende novamente).

Vamos ver, agora, como ler este arquivo. Digite o programa de figura 48.3

Figura 48.3

```
100 OPEN"A:LISTA.SEQ" FOR INPUT AS #1
110 IF EOF(1) THEN GOTO 150
120 INPUT #1,A$
130 N=N+1:PRINT N,A$
140 GOTO 110
150 CLOSE:END
```

A linha 100 abre o mesmo arquivo só que agora para leitura (FOR INPUT) . Para ler os dados, ao invés do PRINT# usamos o INPUT# (linha 120).

Se durante a leitura a função EOF (1) (End of File do arquivo 1) for verdadeiro, assumindo o valor -1(veja dica 38) , o programa é desviado para seu final (linha 150). Caso contrário ele retorna à leitura (GOTO 110).

Rodando o programa, você deve obter a tela da figura 48.4. O contador N foi colocado para você contar o número de registros contidos no arquivo.

Figura 48.4

```
RUN
 1              ANA 240-8866
 2              SOLANGE 543-1147
 3              SUZY 64-7898
 4              JACQUELINE 290-9169
 5              FLAVIA 240-1131
Ok
█
```

Vamos agora abrir um arquivo randômico que leia este sequêncial byte a byte.

Note que, para abrir um arquivo randômico, não precisamos mais especificar se é para escrita (FOR OUTPUT) ou para leitura (FOR INPUT).

Basta especificar o comprimento do registro (LEN) que no caso é 1, pois queremos ler o arquivo byte a byte.

Para diferênciar escrita de leitura, no arquivo rondômico, usamos respectivamente o:

PUT# **nº do arquivo, nº do registro**

e

GET# **nº do arquivo, nº do registro**

Digite o programa de figura 48.5 e veja seu efeito antes de prosseguir nesta explicação.

Figura 48.5

```
100 OPEN"A:LISTA.SEQ" AS #1 LEN=1
110 FIELD #1,1 AS A$
120 SCREEN 1
130 N=N+1
140 GET#1,N
150 A= ASC(A$)
160 IF A<32 THEN PRINT CHR$(1)+CHR$(64+A)
; ELSE PRINT CHR$(A);
170 IF A=10 THEN PRINT
180 IF A=26 THEN END
190 GOTO 130
```

A instrução FIELD da linha 110 define quantos campos e quais seus comprimentos. Em cada registro de LEN=1 podemos, obviamente, definir um único campo (A$) de comprimento unitário!

A linha 160 permite a impressão de cada caracter lido, mesmo de código menor que 32 (veja dica 26).

Você deverá obter a tela da figura 48.6

Figura 48.6

```
ANA 240-8866♪█
SOLANGE 543-1147♪█
SUZY 64-7898♪█
JACQUELINE 290-9169♪█
FLAVIA 240-1131♪█
└
Ok
█
```

Os caracteres "estranhos" que aparecem são os "separadores de registro" do arquivo sequêncial original (LISTA.SEG) .

Eles têm o significado explicado na figura 48.7:

Figura 48.7

```
♪ =CHR$(13)=CR  | SEPARADORES
                ▶      DE
█ =CHR$(10)=LF  |   REGISTROS

┌ =CHR$(26)=SUB ▶FIM DE ARQUIVO
```

Foi por esta razão que a linha 170 dá um PRINT ao encontrar o CHR$(10), obrigando um salto de linha. Da mesma forma, a linha 180 encerra o programa ao encontrar o CHR$(26) (Fim de Arquivo).

O programa da figura 48.5 foi estruturado desta forma por motivos didáticos.

Na realidade estes 3 caracteres de contrôle se encarregam de fazer voltar o cursor (no PRINT) ou a cabeça de impressão (no LPRINT) , pular a linha e detectar o fim do arquivo.

Experimente digitar a versão simplificada na listagem 48.8

Figura 48.8

```
100 OPEN"A:LISTA.SEQ" AS #1 LEN=1
110 FIELD #1,1 AS A$
130 N=N+1
140 GET#1,N
160 PRINTA$
190 GOTO 130
```

você vai obter a tela da figura 48.9 e uma mensagem indicando onde se deu o fim do arquivo .

Figura 48.9

```
RUN
ANA 240-8866
SOLANGE 543-1147
SUZY 64-7898
JACQUELINE 290-9169
FLAVIA 240-1131
Fim do arquivo em 140
Ok
█
```

O arquivo sequêncial tem uma grande vantagem em relação ao randômico pois economiza espaço em disco: afinal cada registro tem o comprimento exato (+ 2 bytes) que precisa.

Em compensação é extremamente lento buscar nele um dado registro. No randomico gasta-se mais memória, mas o manuseio é extremamente mais simples. Digamos que você comprou um drive recentemente mas tem uma série de arquivos sequenciais em fita.

Vejamos como transformar um arquivo sequêncial num arquivo rondômico.

Digite o programa de figura 48.10 e rode-o para ver como funciona

Figura 48.10

```
100 SCREEN 0:WIDTH 40:KEY OFF
110 MAXFILES=2
120 OPEN"A:LISTA.SEQ" FOR INPUT AS #1
130 OPEN"A:LISTA.RAN" AS #2 LEN=20
140 FIELD #2,20 AS R$
150 IF EOF(1) THEN 210
160 INPUT #1,S$
170 N=N+1
180 LSET R$=S$
190 PUT #2,N
200 GOTO 150
210 FOR I=1 TO 2
220 PRINT "ARQUIVO";I;LOF(I);"BYTES"
230 NEXT I
240 CLOSE:END
```

A linha 110 define o número de arquivos que serão abertos e as 120 e 130 abrem, respectivamente um sequencial para leitura (já existente no disco) e um rondômico com registros de 20 bytes (se não a Jacqueline não cabe!). A linha 140 define um único campo (R$) de 20 bytes. A linha 150 detecta o fim do arquivo e a 160 lê os dados contidos no arquivo sequêncial. A linha 170 atualiza o contador e a 180 atribui à variável R$ o conteúdo de S$, alinhando pela esquerda (LSET).

Se quiséssemos alinhar pela direita, usaríamos o RSET.

A linha 190 escreve os dados no arquivo rondômico (#2).

O laço 210-230 fornece o número de bytes de cada arquivo. Note que, como temos 5 registros de 20 bytes cada, o arquivo randômico tem 100 bytes, enquanto que o sequencial ocupa 85.

Agora que você tem no disco o arquivo "LISTA.RAN", digite o programa da figura 48.11 pára ver como é fácil manusear um arquivo randômico.

Se você absorveu o que foi dito até agora, entenderá o conteúdo do programa.

Atente apenas para a linha 140: como descobrir o

número de registros de um arquivo randômico? Basta calcular o LOF (nº de bytes) dividido pelo comprimento do registro (no caso, 20).

Agora cabe a você transformar ou criar seu próprios arquivos. Se você quiser informações mais detalhadas sobre este assunto aconselhamos a leitura o "DRIVES LEOPARD de 3 1/2" que, apesar do nome, serve também para quem tem 5 1/4 !

Figura 48.11

```
100 SCREEN 0:KEYOFF
110 OPEN"A:LISTA.RAN" AS #1 LEN=20
120 FIELD #1,20 AS R$
130 PRINT "ARQUIVO";1;LOF(1);"BYTES"
140 PRINT:R=LOF(1)/20
150 PRINT R;"REGISTROS"
160 PRINT:PRINT"P/TERMINAR DIGITE F"
170 PRINT "QUAL REGISTRO ( 1/";R;")
180 A$=INPUT$(1):A=VAL(A$)
190 IF A$="F" OR A$="f" THEN 270
200 IF A<1 OR A>R THEN 180
210 GET #1,A
220 PRINT TAB(11);STRING$(22,&HDC)
230 PRINT "REGISTRO";A;CHR$(&HDD);R$;
240 PRINT CHR$(&HDB)
250 PRINT TAB(11);STRING$(22,&HDF)
260 GOTO 170
270 CLOSE:END
```

49 USANDO A EXPANSÃO DE MEMóRIA

O MSX foi planejado para um mínimo de 8 Kbytes de memória RAM. Os micros nacionais já veêm com 64 Kbytes, o que permite o uso do MSX-DOS.

A memória pode ser expandida até 1 Megabyte, tanto memória RAM como ROM, dependendo apenas da vontade do usuário.

Existem expansões de memória de 64 Kbytes à venda, mas está memória não está disponível diretamente para o BASIC.

Portanto, se você comprar uma expansão e a mensagem inicial do fabricante não apresentar mais memória livre como seria de se esperar, não se desespere, pois NÃO É DEFEITO DA EXPANSÃO!

O máximo de memória livre para o BASIC são os 28815 bytes.

Os 64 Kbytes da expansão ficam "em paralelo" com a RAM e com a ROM do micro. Consulte o "APROFUNDANDO-SE NO MSX" para conhecer melhor essa estrutura.

Para gerenciar as páginas de memória existem duas rotinas do BIOS. Uma lê um determinado endereço em um slot e a outra ecreve um dado no endereço de um slot. São elas:

RDSLT (&H000C) para leitura
WRSLT (&H0014) para escrita

Elas funcionam analogamente ao comando PEEK e POKE do BASIC, mas só podem ser acessadas em Linguagem de Máquina.

O programa da figura 49.1 apresenta duas rotinas para que você possa, trabalhando em BASIC, ler e escrever em outros slots.

Figura 49.1

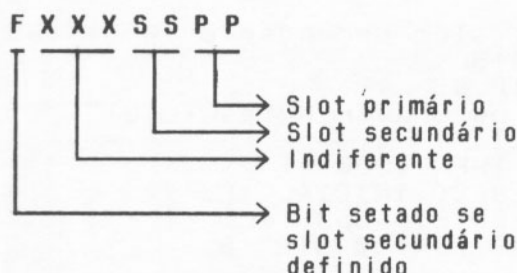
```
1000 DATA 3A,1B,D0,2A,1C,D0,CD,0C
1010 DATA 00,32,1E,D0,C9,3A,1B,D0
1020 DATA 2A,1C,D0,ED,5B,1E,D0,CD
1030 DATA 14,00,C9,00,00,00,00
1040 FOR L=&HD000 TO &HD01E
1050 READ A$
1060 POKE L,VAL("&H"+A$)
```

```
1070 NEXT
1080 DEFUSR=&HD000
1090 DEFUSR1=&HD00D
1100 INPUT"Qual o slot (0-3)";SL
1110 INPUT"secundário (s/n)";S$
1120 IF S$="S"OR S$="s" THEN FL=&H80 ELSE
 1140
1130 INPUT"Qual slot secundário (0-3)";SS
1140 BS=FL+SS*4+SL
1150 POKE &HD01B,BS
1160 PRINT"Ler ou Gravar bytes (L/G)"
1170 A$=INPUT$(1)
1180 IF A$="G" THEN 1320
1190 INPUT"ENDEREÇO INICIAL";EI
1200 IF EI<0 THEN EI=65536!+EI
1210 FOR L=EI TO EI+100 STEP 8
1220 R=0
1230 PRINTRIGHT$("000"+HEX$(L+R),4)" ";
1240 FOR R=0 TO 7
1250 POKE &HD01D,INT((L+R)/256)
1260 POKE &HD01C,L+R-256*INT((L+R)/256)
1270 POKE USR(0),0
1280 PRINTRIGHT$("0"+HEX$(PEEK(&HD01E)),2
)" ";
1300 NEXT R:PRINT:NEXT L
1310 GOTO 1100
1320 INPUT"ENDEREÇO INICIAL";EI
1330 IF EI<0 THEN EI=65536!+EI
1340 FOR L=EI TO 65535! STEP 8
1350 R=0
1360 PRINTRIGHT$("000"+HEX$(L+R),4)" ";
1370 FOR R=0 TO 3
1380 POKE &HD01D,INT((L+R)/256)
1390 POKE &HD01C,L+R-256*INT((L+R)/256)
1400 POKE USR(0),0
1410 PRINTRIGHT$("0"+HEX$(PEEK(&HD01E)),2
)"-";
1420 H$=INPUT$(2):PRINTH$" ";
1430 IF H$="  " THEN PRINT:GOTO 1100
1440 POKE &HD01E,VAL("&H"+H$)
1450 POKE 0,USR1(0)
1460 NEXT R:PRINT:NEXT L
1470 GOTO 1160
```

As linhas de 1000 a 1070 instalam a rotina a partir do endereço &HD000.

A rotina de leitura pede, no endereço &HD01B

número do SLOT, que deve ser montado a partir d número, em binário, do slot primário e do secundário, conforme os bits mostrados na figura 49.2

Figura 49.2

No endereço &HD01C e &HD01D deve ser POKEado (na forma LSB e MSB) o endereço do SLOT que queremos consultar.

O endereço &HD01E conterá o valor lido (se chamada a rotina de leitura) ou deverá conter o valor a ser POKEado em outro slot (para a rotina de escrita)

O ponto de entrada dessas duas rotinas está nos endereços:

&HD000 Leitura
&HD00D Escrita

O restante do programa, é um monitor para que você possa "fuçar" pela memória do micro!

Lembre-se que:

Para Ler um valor em qualquer slot com essa rotina você deve usar os comandos:

```
DEFUSR=&HD000
POKE &HD01B, numero do slot
POKE &HD01C, LSB do endereço
POKE &HD01D, MSB do endereço
POKE 0,USR(0)
```

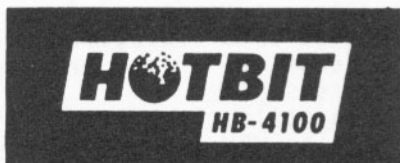
após esses comandos, o valor estará copiado n endereço &HD01E.

Para escrever um valor no slot desejado, comande:

```
DEFUSR=&HD00D
```

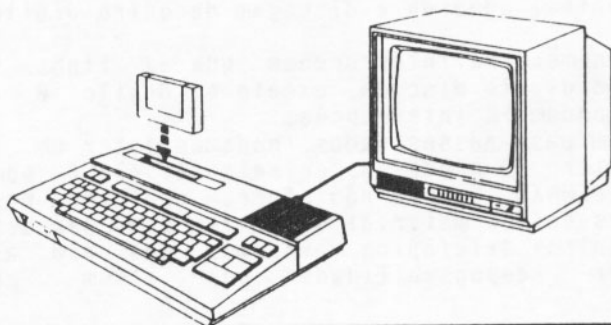
```
POKE &HD01B, numero do slot
POKE &HD01C, LSB do endereço
POKE &HD01D, MSB do endereço
POKE &HD01E, valor a ser escrito
POKE 0,USR(0)
```

Você pode usar, também, essas duas rotinas para acessar os 32 Kbytes que ficam "escondidos" pela ROM quando estamos usando o BASIC.

INSTALAÇÃO

Os procedimentos abaixo referem-se a instalação do produto:

- O computador HOTBIT deve estar desligado;

- O cartucho deve ser colocado em um dos slots, observando a seta que indica a posição correta do mesmo (figura).

50 DISCADOR AUTOMATICO COM O MSX

Uma aplicação muito poderosa para os computadores pessoais é a automação de aparelhos elétricos. No Exterior existem "kits" que são ligados a um microcomputador e estes passam a controlar pequenos braços-robôs, barcos, ferrovias de brinquedo e mesmo eletrodomésticos, luzes e telefones. No Brasil a realidade é outra e os micros pessoais quase sempre são usados como videogame. Só usuários mais audaciosos se atrevem a criar programas realmente bons ou periféricos.

Uma aplicação muito interessante e facilmente aplicada ao MSX é um discador automático de telefone. Um programa em BASIC e o relê de controle do cassete do micro são o que bastam.

O processo de discar no telefone é o seguinte:

1: O fone está no gancho e o circuito com a linha está aberto.

2: O fone é tirado do gancho; o circuito é fechado e a central espera um número ser discado.

3: Um dígito é discado. Por exemplo, 5.

4: O circuito é interrompido durante 1/20 segundo e fechado durante outro 1/20 segundo (total 1/10 segundo). Este processo é repetido 5 vezes, para dígito 5.

5: Após o dígito, o circuito fica 1/5 segundo fechado para informar à central que este dígito terminou.

6: A central aguarda a discagem de outro dígito.

O número de interrupções que a linha sofre é igual ao dígito discado, exceto o dígito 0 ao qual correspondem 10 interrupções.

Com base nesses dados, podemos fazer um discador com o micro. O processo, entretanto, não é homologado pela TELEBRÁS. Embora não ofereça qualquer perigo de dano pessoal ou material e nem tem como ser descoberto pela central telefônica, avisamos que não assumimos qualquer responsabilidade por algum problema.

Repetimos que a adaptação é totalmente segura.

Corte um dos fios da linha telefônica e instale um jaque fêmea pequeno (tipo REMOTE de gravador) entre os pontos separados. Em seguida, conecté o plugue preto (mais fino) do cabo do cassete. Observe o esquema de ligação na figura 50.1

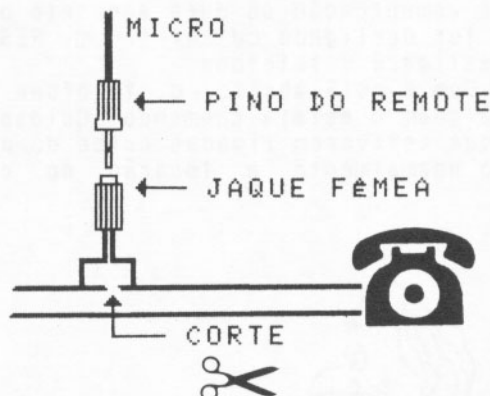
Figura 50.1

Após verificar as ligações, digite o programa da figura 50.2

Figura 50.2

```
10 CLS:MOTOR OFF
20 INPUT "Numero a discar ";N$
30 MOTOR OFF:FOR I=1 TO 500:NEXT I
40 MOTOR ON:FOR I=1 TO 500:NEXT I
50 FOR I=1 TO LEN(N$)
60 A=ASC(MID$(N$,I))-48
70 IF A=0 THEN A=10
80 FOR J=1 TO A
90 MOTOR OFF:FOR K=1 TO 20:NEXT K
100 MOTOR ON:FOR K=1 TO 20:NEXT K
110 NEXT J
120 FOR J=1 TO 200:NEXT J
130 NEXT I
140 I$=INPUT$(1)
150 IF I$="r" OR I$="R" THEN 30
160 IF I$="n" OR I$="N" THEN 10
170 GOTO 140
```

Para discar um número, retire o fone do gancho (o telefone deverá estar mudo, é normal) e rode o

programa. Entre o número desejado. O telefone deverá apresentar o tom de discar e logo começará a discagem automática.

Se o número der ocupado, tecle "R" para discar de novo o mesmo número. Para discar um número diferente, tecle N. E boa conversa!!!

Obs: A comunicação só dura enquanto o micro ficar ligado. Se for desligado ou sofrer um RESET, o relê abrirá e desligará o telefone.

Obs2: Com o relê aberto, o telefone não tocará mesmo que alguém o esteja chamando. Cuidado, então. As extensões que estiverem ligadas antes do ponto cortado funcionarão normalmente e tocarão no caso de uma chamada.

APÊNDICE I

LISTAGENS ASSEMBLER

Apresentamos, a seguir, a lintagens dos programas fontes das dicas que utilizam o recurso da Linguagem de Máquina.

Recomendamos a leitura prévia da dica referente ao programa para uma melhor compreensão.

Todos os programas, foram feitos e compilados usando o ASMCOCAR. Se você possui um compilador com outra sintaxe preste atenção durante a transferência.

14

```
1000 LPTOUT:        EQU 000A5H
1010 HLPT:          EQU 0FFB6H
1020      ORG 0D000H
1030 INSTAL:                 ;Altera hook HLPT
1040   LD HL,INICIO          ;para o inicio do
1050   LD (HLPT+1),HL        ;programa.
1060   LD A,0C3H             ;
1070   LD (HLPT),A           ;
1080   RET                   ;
1090 INICIO:                 ;Compara com o zero.
1100   CP '0'                ;Se diferente retorna
1110   RET NZ                ;
1120   LD A,'/'              ;Envia a barra.
1130   CALL LPTOUT           ;
1140   LD A,08               ;Retorna o carro.
1150   CALL LPTOUT           ;
1160   LD A,'O'              ;Envia a letra O.
1170   CALL LPTOUT           ;
1180   INC SP                ;Descarta stack.
1190   INC SP                ;
1200   RET                   ;Retorna.
1210   END
```

15

```
1000 HLPTO:       EQU 0FFB6H
1010 LPTOUT:      EQU 000A5H
1020        ORG 0D000H
1030    LD HL,INICIO      ;Muda hook HLPT
1040    LD (HLPTO+1),HL   ;para o inicio do
1050    LD A,0C3H         ;programa.
1060    LD (HLPTO),A      ;
1070    LD A,0FFH         ;Muda filtro BRASCII
1080    LD (0F417H),A     ;
1090    RET               ;
1100 INICIO:              ;
1110      LD (COD),A      ;Armazena caracter
1120      EXX             ;Preserva registros
1130      LD HL,HLPTO     ;Muda hook HLPT.
1140      LD (HL),0C9H    ;
1150      LD A,(COD)      ;
1160      CP 80H          ;Verifica se e' le-
1170      JR C,FIM        ;tra acentuada
1180     SUB 80H          ;Converte codigo
1190     LD L,A           ;no elemento da tabe-
1200     LD H,0           ;la de conversao.
1210     ADD HL,HL        ;
1220     LD DE,TABCOV     ;
1230     ADD HL,DE        ;
1240     LD A,(HL)        ;Envia letra para a
1250     CALL LPTOUT      ;impressora.
1260     LD A,8           ;Retrocede o carro.
1270     CALL LPTOUT      ;
1280     INC HL           ;Avana na tabela de
1290     LD A,(HL)        ;conversao.
1300     CALL LPTOUT      ;envia p/ impressora.
1310     POP HL           ;
1320 FIM:                 ;
1330     LD HL,HLPTO      ;Aponta hook HLPT
1340     LD (HL),0C3H     ;para o programa.
1350     EXX              ;Restaura registros.
```

```
1360     RET                  ;Retorna ao interpre-
1370                          ;tador BASIC
1380 COD:         DEFB 0
1390 TABCOV: DEFB  043H,02CH,075H,022H
1400 DEFB  065H,027H,061H,05EH
1410 DEFB  041H,027H,060H,061H
1420 DEFB  022H,000H,063H,02CH
1430 DEFB  065H,05EH,049H,027H
1440 DEFB  04FH,05EH,055H,027H
1450 DEFB  041H,05EH,045H,05EH
1460 DEFB  04FH,05EH,041H,060H
1470 DEFB  045H,027H,000H,000H
1480 DEFB  000H,000H,06FH,05EH
1490 DEFB  000H,000H,060H,06FH
1500 DEFB  05EH,075H,000H,000H
1510 DEFB  000H,000H,000H,000H
1520 DEFB  000H,055H,0FFH,000H
1530 DEFB  000H,000H,000H,000H
1540 DEFB  000H,000H,000H,000H
1550 DEFB  061H,027H,069H,027H
1560 DEFB  06FH,027H,075H,027H
1570 DEFB  000H,000H,000H,000H
1580 DEFB  061H,05FH,06FH,05FH
1590 DEFB  000H,000H,000H,000H
1600 DEFB  000H,000H,000H,000H
1610 DEFB  000H,000H,000H,000H
1620 DEFB  000H,000H,000H,000H
1630 DEFB  041H,07EH,061H,07EH
1640 DEFB  049H,07EH,069H,07EH
1650 DEFB  04FH,07EH,06FH,07EH
1660 DEFB  055H,07EH,075H,07EH
1670 END
```

27

```
1000 ;--------------------------------
1010 ; Fontes de caracteres na VRAM
1020 ;--------------------------------
```

```
1030          ORG    0C000H
1040 ;--------------------------------
1050 ; Rotinas do BIOS
1060 ;--------------------------------
1070  INITXT:         EQU 0006CH
1080  INIT32:         EQU 0006FH
1090  RDVRM:          EQU 0004AH
1100  WRTVRM:         EQU 0004DH
1110 ;--------------------------------
1120 ; Variaveis do Sistema
1130 ;--------------------------------
1140  TXTCGP:         EQU 0F3B7H
1150  T32CGP:         EQU 0F3C1H
1160  VALTYP:         EQU 0F663H
1170  DAC:            EQU 0F7F6H
1180 ;--------------------------------
1190 ; Recebe parametro do BASIC
1200 ;--------------------------------
1210     CP 2
1220     RET NZ
1230     INC HL
1240     INC HL
1250     LD A,(HL)
1260     CP 1
1270     JR NZ,ITALI1
1280 ;--------------------------------
1290 ; Fonte de caracteres 3x6
1300 ;--------------------------------
1310  COL64:
1320     CALL SUB1
1330     LD BC,2048
1340     LACO0:
1350     PUSH BC
1360     PUSH HL
1370     CALL RDVRM
1380     LD D,A
1390     AND 00011000B
1400     LD B,A
1410     LD A,D
1420     AND 11000000B
```

```
1430       LD C,A
1440       LD A,D
1450       AND 00100000B
1460       LD D,A
1470       LD A,B
1480       CP 0
1490       JR Z,DSV1
1500       LD B,00100000B
1510   DSV1:
1520       LD A,C
1530       CP 0
1540       JR Z,DSV2
1550       LD C,10000000B
1560   DSV2:
1570       LD A,D
1580       CP 0
1590       JR Z,DSV3
1600       LD D,01000000B
1610   DSV3:
1620       LD A,B
1630       OR C
1640       OR D
1650       CALL WRTVRM
1660       POP HL
1670       POP BC
1680       INC HL
1690       DEC BC
1700       LD A,B
1710       OR C
1720       JR NZ,LACO0
1730       RET
1740  ;------------------------------
1750  ; Fonte ITALICO I
1760  ;------------------------------
1770   ITALI1:
1780       CP 2
1790       JR NZ,ITALI2
1800       CALL SUB1
1810   LACO1:
1820       CALL     SUB3
```

```
1830  LAC02:
1840      PUSH HL
1850      ADD  HL,BC
1860      CALL RDVRM
1870      LD D,A
1880      LD A,C
1890      CP 4
1900      JR NC,DSV4
1910      XOR A
1920      LD A,D
1930      RRCA
1940      CALL WRTVRM
1950  DSV4:
1960      POP HL
1970      INC BC
1980      LD A,C
1990      CP 8
2000      JR NZ,LAC02
2010      CALL SUB5
2020      JR NZ,LAC01
2030      RET
2040 ;-------------------------------
2050 ; Fonte ITALICO 2
2060 ;-------------------------------
2070  ITALI2:
2080      CP 3
2090      JR NZ,ITALI3
2100      CALL SUB2
2110  LAC03:
2120      CALL SUB3
2130  LAC04:
2140      PUSH HL
2150      ADD HL,BC
2160      CALL RDVRM
2170      LD D,A
2180      LD A,C
2190      CP 1
2200      JR NC,DSV5
2210      XOR A
2220      LD A,D
```

```
2230      RRCA
2240      RRCA
2250      RRCA
2260      JR DSV8
2270  DSV5:
2280      CP 2
2290      JR NC,DSV6
2300      XOR A
2310      LD A,D
2320      RRCA
2330      RRCA
2340      RRCA
2350      JR DSV8
2360  DSV6:
2370      CP 3
2380      JR NC,DSV7
2390      XOR A
2400      LD A,D
2410      RRCA
2420      RRCA
2430      JR DSV8
2440  DSV7:
2450      CP 4
2460      JR NC,DSV9
2470      XOR A
2480      LD A,D
2490      RRCA
2500   DSV8:
2510      CALL WRTVRM
2520   DSV9:
2530      POP HL
2540      INC BC
2550      LD A,C
2560      CP 8
2570      JR NZ,LACO4
2580      CALL    SUB5
2590      JR NZ,LACO3
2600      RET
2610 ;------------------------------
2620 ; Fonte ITALICO 3
```

```
2630 ;------------------------------
2640  ITALI3:
2650     CP 4
2660     JR NZ,BOLD
2670     CALL SUB1
2680  LAC05:
2690     CALL SUB3
2700  LAC06:
2710     PUSH HL
2720     ADD HL,BC
2730     CALL RDVRM
2740     LD D,A
2750     XOR A
2760     LD A,D
2770     RRCA
2780     LD D,A
2790     LD A,C
2800     CP 4
2810     LD A,D
2820     JR NC,DSVA
2830     XOR A
2840     LD A,D
2850     RLCA
2860  DSVA:
2870     CALL SUB4
2880     JR NZ,LAC06
2890     CALL SUB5
2900     JR NZ,LAC05
2910     RET
2920 ;------------------------------
2930 ; Fonte de caracteres BOLD
2940 ;------------------------------
2950  BOLD:
2960     CP 5
2970     RET NZ
2980     CALL SUB2
2990  LAC07:
3000     CALL SUB3
3010  LAC08:
3020     PUSH HL
```

```
3030      ADD HL,BC
3040      CALL RDVRM
3050      LD D,A
3060      XOR A
3070      LD A,D
3080      RRCA
3090      OR D
3100      CALL SUB4
3110      JR NZ,LAC08
3120      CALL    SUB5
3130      JR NZ,LAC07
3140      RET
3150 ;-----------------------------
3160 ; SUB-ROTINA 1 | p/ SCREEN 0
3170 ;-----------------------------
3180  SUB1:
3190     CALL INITXT
3200     LD A,(TXTCGP)
3210     LD L,A
3220     INC HL
3230     LD A,(TXTCGP+1)
3240     LD H,A
3250     LD BC,0
3260     RET
3270 ;-----------------------------
3280 ; SUB-ROTINA 2 | p/ SCREEN 1
3290 ;-----------------------------
3300  SUB2:
3310     CALL INIT32
3320     LD A,(T32CGP)
3330     LD L,A
3340     LD A,(T32CGP+1)
3350     LD H,A
3360     LD BC,0
3370     RET
3380 ;-----------------------------
3390 ; SUB-ROTINA 3 | HL <= HL+8xBC
3400 ;-----------------------------
3410  SUB3:
3420     POP DE
```

```
3430    PUSH BC
3440    PUSH HL
3450    PUSH DE
3460    ADD HL,BC
3470    ADD HL,BC
3480    ADD HL,BC
3490    ADD HL,BC
3500    ADD HL,BC
3510    ADD HL,BC
3520    ADD HL,BC
3530    ADD HL,BC
3540    LD BC,0
3550    RET
3560 ;------------------------------------
3570 ; SUB-ROTINA 4 | laco de 8 vezes
3580 ;------------------------------------
3590  SUB4:
3600    CALL WRTVRM
3610    POP DE
3620    POP HL
3630    PUSH DE
3640    INC BC
3650    LD A,C
3660    CP 8
3670    RET
3680 ;------------------------------------
3690 ; SUB-ROTINA 5 | laco de 255
3700 ;------------------------------------
3710  SUB5:
3720    POP DE
3730    POP HL
3740    POP BC
3750    PUSH DE
3760    INC BC
3770    LD A,B
3780    CP 1
3790    RET
3800    END
```

31

```
1000 HKEYC:                    EQU 0FDCCH
1010 KEYBUF:                   EQU 0FBF0H
1020 PUTPNT:                   EQU 0F3F8H
1030 GETPNT:                   EQU 0F3FAH
1040 KILBUF:                   EQU 00156H
1050 CHGET:                    EQU 0009FH
1060      ORG 0D000H
1070    LD HL,INICIO       ;Muda HOOK HKEYC
1080    LD (HKEYC+1),HL
1090    LD A,0CDH
1100    LD (HKEYC),A
1110    RET
1120 INICIO:
1130    CP 03EH            ;Compara com SELECT
1140    RET NZ             ;Retorna se diferente
1150    PUSH HL            ;Armazena registradore
                            s
1160    PUSH BC            ;
1170    PUSH DE            ;
1180    PUSH AF            ;
1190    CALL KILBUF        ;Limpa o buffer
1200    CALL CHGET         ;Aguarda tecla
1210    CP 03AH            ;Transforma codigo
1220    JR C, J1           ;em nibble mais
1230    SUB 7              ;significativo
1240 J1:                   ;
1250    SUB 30H            ;
1260    SLA A              ;
1270    SLA A              ;
1280    SLA A              ;
1290    SLA A              ;
1300    AND 0F0H
1310    LD D,A
1320    CALL CHGET         ;Aguarda nova tecla
1330    CP 03AH            ;Transforma codigo
1340    JR C, J2           ;ASCII em nibble
```

```
1350   SUB 7              ;menos significativo
1360 J2:
1370   SUB 30H
1380   OR D               ;Junta dois nibbles
1390   LD HL,KEYBUF       ;Aponta GETPNT para o
1400   LD (GETPNT),HL     ;inicio do buffer
1410   CP 20H             ;verifica se e caracte
                           r
1420   JR C,CMINUS        ;grafico menor que 32
1430   LD (HL),A          ;armazena codigo no bu
                           ffer
1440   INC HL             ;Aponta PUTPNT para o
1450   LD (PUTPNT),HL     ;inicio do buffer+1
1460   JR FIM             ;Finaliza.
1470 CMINUS:              ;Insere caracter 01
1480   LD D,A             ;e codigo do carac-
1490   LD A, 01H          ;ter mais 64 no
1500   LD (HL),A          ;buffer.
1510   INC HL             ;
1520   LD A, 040H         ;
1530   ADD A,D            ;
1540   LD (HL),A          ;
1550   INC HL             ;
1560   LD (PUTPNT),HL     ;
1570 FIM:                 ;
1580   POP AF             ;Retorna valores
1590   POP DE             ;dos registradores.
1600   POP BC             ;
1610   POP HL             ;
1620   RET                ;Retorna da HOOK.
1630 END
```

32

```
1000 HKEYC:                  EQU 0FDCCH
1010 KEYBUF:                 EQU 0FBF0H
1020 PUTPNT:                 EQU 0F3F8H
```

```
1030 GETPNT:               EQU 0F3FAH
1040 KILBUF:               EQU 00156H
1050 CHGET:                EQU 0009FH
1060     ORG 0C000H
1070   LD HL,INICIO     ;Muda HOOK HKEYC
1080   LD (HKEYC+1),HL;para o inicio
1090   LD A,0C3H        ;do programa.
1100   LD (HKEYC),A     ;
1110   RET              ;
1120 INICIO:            ;
1130   CP 030H          ;Compara com codigo
1140   JR Z, LIGSHIFT ;da SHIFT
1150   CP 048H          ;Verifica se tecla
1160   JR C, DESHIFT  ;do num reduzido.
1170   PUSH DE          ;Armazena registra-
1180   PUSH BC          ;dores.
1190   PUSH HL          ;
1200   PUSH AF          ;
1210   LD A,(FLSHIFT) ;Verifica se SHIFT
1220   CP 0FFH          ;foi pressionada.
1230   JR Z,INVERT      ;liga ou desliga hexa
1240   LD A,(FLONOFF) ;Verifica se teclado
1250   CP 0             ;hexa esta ligado.
1260   JR Z,TADESL      ;
1270   POP AF           ;Converte para novo
1280   LD E,A           ;codigo.
1290   LD A,061H        ;
1300   SUB E            ;
1310 FIM:               ;
1320   POP HL           ;Retorna registradores
1330   POP BC           ;
1340   POP DE           ;
1350   RET              ;
1360 DESHIFT:           ;
1370   PUSH AF          ;Armazena codigo.
1380   LD A,0           ;Zera flag.
1390   LD (FLSHIFT),A ;
1400   LD A,(FLONOFF) ;Testa flag de liga-
1410   CP 0             ;desliga.Se desligado
1420   JR Z,FIM2        ;retorna.
```

```
1430    POP AF          ;
1440    PUSH AF         ;
1450    CP 0BH          ;compara c/ codigo
1460    JR Z,TCLIGUAL   ;da tecla igual.
1470    CP 013H         ;Compara c/ codigo
1480    JR Z,TCLPNTDC   ;da tecla do ponto
1490 FIM2:              ;
1500    POP AF          ;retorna da hook
1510    RET             ;
1520 TCLIGUAL:          ;Converte para
1530    POP AF          ;novo codigo
1540    XOR A           ;
1550    LD C,017H       ;
1560    RET             ;
1570 TCLPNTDC:          ;Converte para
1580    POP AF          ;novo codigo
1590    LD C,016H       ;
1600    RET             ;
1610 TADESL:            ;
1620    POP AF          ;Retorna da hook
1630    JR FIM          ;
1640 INVERT:            ;
1650    LD A,0          ;ZERA flag da tecla
1660    LD (FLSHIFT),A  ;SHIFT
1670    LD A,(FLONOFF)  ;INVERTE flag de
1680    CPL             ;ligado/desligado
1690    LD (FLONOFF),A  ;
1700    POP AF          ;Retorna valores dos
1710    POP DE          ;registradores.
1720    POP BC          ;
1730    POP HL          ;
1740    RET             ;
1750 LIGSHIFT:          ;
1760    PUSH AF         ;SETA flag ( SHIFT
1770    LD A,0FFH       ;pressionada)
1780    LD (FLSHIFT),A  ;
1790    POP AF          ;
1800    RET             ;
1810 FLSHIFT:  DB 0     ;
1820 FLONOFF:  DB 0     ;
1830 END
```

49

```
1000 RDSLT:          EQU 0000CH
1010 WRSLT:          EQU 00014H
1020      ORG 0D000H
1030    LD   A,(SLOT)         ;Carrega va-
1040    LD   HL,(ENDERECO)    ;lores.
1050    CALL RDSLT            ;Chama rotina
1060    LD   (VALOR),A        ;Armazena na
1061                          ;memoria.
1070    RET                   ;Retorna.
1080    LD   A,(SLOT)         ;Carrega va-
1090    LD   HL,(ENDERECO)    ;lores.
1100    LD   DE,(VALOR)       ;
1110    CALL WRSLT            ;Chama rotina
1120    RET                   ;Retorna.
1130 SLOT:       DB 0
1140 ENDERECO:   DW 0
1150 VALOR:      DW 0
1160 END
```

APÊNDICE II

MAPA DA VRAM

```
SCREEN 0:
BASE(0)  = 00000 = POSIÇÃO DOS CARACTERES NA TELA
BASE(2)  = 02048 = TABELA DE FORMAÇÃO DOS CARACTERES
SCREEN 1:
BASE(5)  = 06144 = POSIÇÃO DOS CARACTERES NA TELA
BASE(6)  = 08192 = CORES DOS OCTETOS DE CARACTERES
BASE(7)  = 00000 = TABELA DE FORMAÇÃO DOS CARACTERES
BASE(8)  = 06912 = TABELA DE ATRIBUTOS DOS SPRITES
BASE(9)  = 14336 = TABELA DE FORMAÇÃO DOS SPRITES
SCREEN 2:
BASE(10) = 06144 = POSIÇÃO DOS CARACTERES NA TELA
BASE(11) = 08192 = CORES DOS GRUPOS DE 8 PONTOS HORIZ.
BASE(12) = 00000 = TABELA DE FORMAÇÃO DOS CARACTERES
BASE(13) = 06912 = TABELA DE ATRIBUTOS DOS SPRITES
BASE(14) = 14336 = TABELA DE FORMAÇÃO DOS SPRITES
```

SCREEN 3:
BASE(15) = 02048 = POSIÇÃO DOS CARACTERES NA TELA
BASE(17) = 00000 = TABELA DE PADRÕES DE CORES
BASE(18) = 06912 = TABELA DE ATRIBUTOS DOS SPRITES
BASE(19) = 14336 = TABELA DE FORMAÇÃO DOS SPRITES

Impressão e acabamento
(com filmes fornecidos):
EDITORA SANTUÁRIO
Fone (0125) 36-2140
APARECIDA - SP

COLEÇÃO MSX

+5Ø DICAS PARA MSX

Apoiados no sucesso do livro "100 DICAS PARA MSX" continuamos com a nossa proposta inicial de ajudar ao máximo o usuário de MSX.

Todas as dicas contidas neste livro possuem uma explicação dos truques e "mágicas" usados. Todos os programas em Linguagem de máquina estão listados em um apêndice no final do livro.

Foi a maneira encontrada por nossos autores de conciliar os interesses do usuário iniciante (para o qual o BASIC é suficiente) com o do hobbista, cada vez mais sedento de informações que permitam "dissecar" o MSX.